AFRL-RY-WP-TR-2012-0224

VISION FOR TIME-VARYING IMAGES

Richard Granger

Trustees of Dartmouth College

MAY 2012
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for a
purpose other than Government procurement does not in any way obligate the U.S. Gover
The fact that the Government formulated or supplied the drawings, specifications, or othe
does not license the holder or any other person or corporation; or convey any rights or per
to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the USAF 88th Air Base Wing (88 ABW) Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RY-WP-TR-2012-0224 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH THE ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//

DR. PHILIP D. MUMFORD, Project Engineer
Distributed Collaborative Sensor System
Technology Branch

//SIGNED//

KENNETH LITTLEJOHN, Chief
Distributed Collaborative Sensor System
Technology Branch

//SIGNED//

TODD A. KASTLE, Chief
Integrated Electronic and Net-Centric Warfare Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, a
publication does not constitute the Government's approval or disapproval of its ideas or fi

*Disseminated copies will show "//signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
May 2012	Final	19 May 2010 – 20 May 2012

4. TITLE AND SUBTITLE
VISION FOR TIME-VARYING IMAGES

5a. CONTRACT NUMBER
FA8650-10-C-7043

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62304E

6. AUTHOR(S)
Richard Granger

5d. PROJECT NUMBER
2000

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
Y0J3

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Trustees of Dartmouth College
Office of Sponsored Projects
1 Hinman Dartmouth College
Hanover, NH 03755-4099

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYWC

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2012-0224

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
This work was funded in whole or in part by Department of the Air Force contract FA8650-10-C-7043. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government. PAO Case Number: 88ABW-2012-4056, cleared 23 July 2012. Report contains color.

14. ABSTRACT
Visual processing has predominantly been aimed at labeled, static images (e.g., caltech101), ignoring a) moving images, which constitute a vast amount of visual data (e.g., youtube, television, as well as all natural visual real-world experience); and b) unlabeled images; despite the fact that labeling is among the most time-intensive aspect of vision research. We studied 1) the development of tasks for visual processing of moving scenes, to provide the field with datasets and benchmarks, to begin to try to catch up to the very large number of static visual datasets; and 2) development and testing of algorithms for vision for time-varying images (VTV), including evaluation of existing algorithms and development of novel approaches. This grant was intended to be a relatively brief (18 month) initial proof-of-principle effort. It has arguably exceeded its initial aims: we have developed novel algorithms for object recognition and localization in both still images and in videos, and we have carried out initial evaluations comparing the new methods with previous approaches. The results, described herein, are promising, and ongoing work is aimed at extending the initial findings to include a suite of advanced approaches to VTV tasks.

15. SUBJECT TERMS
computer vision, machine learning, machine perception

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Philip D. Mumford
Unclassified	Unclassified	Unclassified	SAR	76	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Table of Contents

Section **Page**

Table of Figures

1. Introduction

Visual processing has predominantly been aimed at labeled, static images (e.g., caltech101), ignoring a) moving images, which constitute a vast amount of visual data (e.g., youtube, television, as well as all natural visual real-world experience); and b) unlabeled images; despite the fact that labeling is among the most time-intensive aspects of vision research. We studied 1) the development of tasks for visual processing of moving scenes, to provide the field with datasets and benchmarks, to begin to try to catch up to the very large number of static visual datasets; and 2) development and testing of algorithms for vision for time-varying images (VTV), including evaluation of existing algorithms and development of novel approaches. This grant was intended to be a relatively brief (18 month) initial proof of principle effort. It has arguably exceeded its initial aims: we have developed novel algorithms for object recognition and localization in both still images and in videos, and we have carried out initial evaluations comparing the new methods with previous approaches. The results, described herein, are promising, and ongoing work is aimed at extending the initial findings to include a suite of advanced approaches to VTV tasks.

2. Novel supervised learning system

The results of this work have been based on algorithms developed from brain circuit analysis, described in part in a series of publications (Rodriguez et al., 2004; Granger 2005; 2006; Felch & Granger 2008; 2011; Granger 2012). In short, multiple regions of the brain perform individual algorithms in isolation, and their combined operation yields a system that takes inputs, constructs memory hierarchies incrementally via learning, and produces suggested output responses. The internal representations are in the form of nested sequences of categories, corresponding to invariant spatiotemporal patterns; these have been analyzed in terms of families of grammars that encode relations organized hierarchically (Granger 2006; 2012).

The corito-striatal loop (CSL) system is one instance of a method that emerges from the interaction of two distinct simpler algorithms (both derived from brain circuit operation): one that performs the operation of unsupervised clustering, and the other performs match-mismatch signaling. The combined system operates in unsupervised mode, except when presented with information that can be used for reinforcement. For instance, complex patterns (e.g., objects with various shapes) may be initially learned via unsupervised relations among their component parts. Whenever these unsupervised representations are found to be at odds with (sparse) supervised information (e.g., when an input is categorized incorrectly), the condition triggers a further unsupervised split of the node in the tree. This successive subdivision repeats until a correct supervised classification is arrived at (Chandrashekar & Granger 2012). The result integrates unsupervised rich representations

via extremely inexpensive methods, with sparse reinforcement signals used when they are available.

The CSL algorithm is a generative method, i.e., it is in the category of algorithms that model data occurring within each presented class, rather than "discriminative" methods, which seek solely to identify differences between classes. Generative models are often taken as performing extra work compared to discriminative models, especially in cases where the only task is to distinguish among labeled classes (Ng & Jordan 2002). The CSL method thus carries out more work than typical classification methods such as support vector machines (SVMs). Yet experiments have been run to compare the algorithms against each other on classification tasks, with surprising results. The classification task is, for the CSL algorithm, a restricted task, since the algorithm is capable of many additional operations (including unsupervised learning, localization, and others); yet this restricted task is among the most widely-used applications in image processing. In this task, the CSL algorithm achieves classification results comparable to those of SVMs, yet uses far less computational cost to do so, despite carrying out the additional work entailed in generative learning (Chandrashekar & Granger 2012).

The CSL mechanism identifies supervised class boundaries as a side-effect of its primary operation, which is that of uncovering structure in the input space independent of supervised labels. It performs solely unsupervised splits of the data into similarity-based clusters. The algorithm, described in detail in Chandrashekar & Granger (2012), is as shown below.

The method constructs a class tree that records unsupervised structure within the data as well as providing a means to perform class prediction on novel samples, as per supervised learning tasks. The PARTITION function denotes an unsupervised clustering algorithm which can in principle be any of a family of clustering routines. The function SUBDIVIDE determines whether or not the data at a given tree node q_n all belong to a single labeled class; if not, the function iterates to further subdivide the node.

This deceptively simple mechanism not only produces a supervised classifier, but also uncovers the similarity structure embedded in the dataset, which competing supervised methods such as SVMs do not do. Despite the fact that competing algorithms, including SVM and Knn methods, were designed expressly to obtain maximum accuracy at supervised classification, we have presented findings indicating that even on this task, the CSL algorithm achieves comparable accuracy while requiring significantly less computational resource cost. This work is described in detail in (Chandrashekar & Granger 2012).

```
Input: Dataset: X = {x_i ∈ R^M} with labels
       Y = {y_i ∈ {1, 2, 3..K}}
Output: Class Tree: A tree rooted at the node
        TRoot
Init: TRoot.X = X, TRoot.Y = Y; TRoot.Labels =
LABELSET(Y)
Q = []; Add(Q, TRoot);
while Q is not empty do
    qn = First node in Q
    if SUBDIVIDE(X_qn, Y_qn) = true then
        [Centroids, Clusters] = PARTITION (X_qn, K*)
        foreach Cluster C_k do
            Node T
            T.X = Clusters[k]
            T.Labels = LABELSET(Y(T.X))
            qn.Branches[k] = Centroids[k]
            qn.Children[k] = T
            Add(Q, T)
        end
    end
end
```

1: CSL Learning Algorithm

3. Joint localization and clustering

Another derived method, JLC, is a generative model that simultaneously identifies the objects in a set of image data and identifies the locus of those objects within the images. The algorithm searches a (preferably very large) dataset and clusters together images containing similar neighboring feature groups, this identifying the occurrence of similar-appearing regions across the images. The method learns the feature histograms (using just a simple bag of features representation) in tandem with the region of the image that contains that feature set; the corresponding region is designated the "foreground" for that object for that image. (Foregrounds can be represented in either of two ways: as bounding boxes or as "superpixels"; the latter are comprised of bottom-up unsupervised segments within the scene.)

The method completely eliminates the need for labeling of images. This is arguably one of the most time consuming and expensive components of image processing. The intuition behind the approach is that objects can be viewed as recurring foreground patterns appearing as coherent image regions. This approach has been used in several other studies such as semantic latent topic models for image clustering (Russel et al., 2006; Fritz & Schiele 2008).

The method is a generative model of "foreground" formation that enables simultaneous image clustering and efficient foreground localization via maximum likelihood estimation. We formulate object discovery as the task of partitioning an unlabeled collection of images into K subsets (clusters) such that all images within each subset share a similar foreground. In order to obtain a method scalable to large collections and many classes, we adopt a

foreground mask-based representation of objects, which enables fast localization given the object model. We do not commit to any particular bottom up segmentation model. Instead we treat the foreground mask as a parameter to be estimated as part of the likelihood optimization. We demonstrate that this leads to localization and image clustering that outperforms competing approaches (Chandrashekar et al., 2012). We view each object instance as a random variable drawn from an unknown distribution common to all instances of that object class. This common distribution assumption constrains all subwindow histograms of an object class to represent subtle variations around a prototypical average histogram. Based on this assumption, our approach poses object discovery as a maximum likelihood estimation problem to be optimized over the collection of unlabeled images. We have presented a method that maximizes this objective by simultaneously solving for the histogram model parameters of the object classes, detecting the object instances of each class in the unlabeled images, and performing a soft semantic clustering of images in the dataset.

4. Object discovery in videos

The work on joint localization and clustering operates on static images, yet most visual input is time-varying input, whether from movies, TV, videos, surveillance, or simply everyday visual experience. Video data actually adds useful constraints to the object recognition task, via inherent temporal consistency across neighboring frames, measurable via a range of optic flow methods.

The video work extends our generative model of static object formation. The method clusters together videos that contain similar objects; here we combine an appearance model as well as a local optic-flow based Markov model into a single objective function defined over the video collection. Since the Markov model is local to a given video, the same object class can be in different movement patterns and yet still contribute to the development of the object class model.

Learning objects from videos has traditionally been attempted in the form of fully-supervised methods, relying on structure from motion or propagating belief by tracking during testing. Indeed, if the task is fully supervised and all video frames are fully annotated, no special methods are required, as each frame is itself a still image. As mentioned, however, generating labels is time consuming and requires expert human intervention; the problem only increases when faced with the multiple frames per second that occur in videos.

5. Related work

Supervised methods that learn to recognize and segment objects in videos include methods that rely on structure from motion (Brostow et al., 2008; Ladick et al., 2010). These methods make assumptions about characteristics of the environment, and even camera angle, that do not readily generalize to real world datasets with unknown camera motion, lighting changes, poor or variable resolution; and moreover, they suffer from the laborious necessity of requiring human hand-labeling. Typical supervised methods for video segmentation are interactive (Bai et al., 2009; Price et al., 2009), requiring input — again time consuming and potentially requiring some expertise — from users.

Unsupervised video segmentation methods include motion segmentation (e.g., Malik & Shi 1998), which cluster pixels in video using bottom-up motion cues; these purely bottom-up methods are highly susceptible to variable camera motion and lighting changes, and are unreliable in certain object motion settings (e.g., when the object starts and stops). Other methods require tracking regions or "keypoints" across frames (Brendel & Todorovic 2009; Brox & Malik 2010; Vasquez-Reina et al., 2010), or formulate clustering objectives to group pixels from all frames using appearance and motion cues (Huang et al., 2009; Grundman et al., 2010). (A model that overcomes some of these drawbacks (Lee, Kim, Grauman, 2011), is set up as a pipeline of arbitrary stages of processing, and its properties have been difficult to characterize.) None of these methods learn any foreground appearance model — i.e., a way of generatively characterizing the learned visual objects.

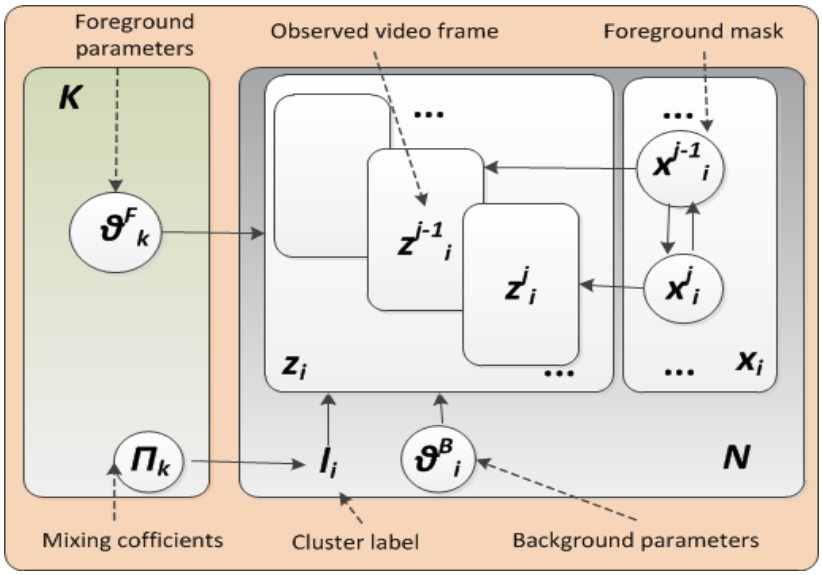
Many approaches have used spatiotemporal feature matching to process video datasets, particularly for gesture recognition (e.g., Dollar et al., 2005; Laptev 2005; Niebles et al., 2008; Willems et al.,2008). It is important to recognize that these methods typically do not generalize to object recognition, in the not-unusual case where there are irregular motions in a video (e.g., an object moving at uneven speed, or stopping and starting). Furthermore, learned spatiotemporal models typically cannot be used to recognize still images, since movement is integrally represented in the learned model.

In contrast, we have taken an approach of simultaneous clustering and localization of objects in unlabeled videos via optimization of a single objective. The method uses both an appearance model and motion model as constraints in the search for object foregrounds in the videos. The learned appearance model operates on still images as well as on the videos from which it was acquired.

6. Generative model for unsupervised object discovery in videos

The figure below illustrates the generative video processing model. We are given a set of *N*

unlabeled videos $z_1,\ldots,z_N$ with each video assumed to contain one of K objects in all of its frames. Let the foreground object content in frame j of video i be described by z_i^j. The twofold objective is to separate the videos into K disjoint subsets (clusters) corresponding to the K object classes, and to localize the object within every frame of each video.



2: Video Processing Model

Let x_i^j denote the (unknown) foreground mask enclosing the object of z_i^j and the foreground mask for the entire video i is then x_i , a sequence of random variables x_i^j. The foreground region corresponding to the mask for the frame is computed as the un-normalized histogram

$$h(z_i^j, x_i^j) \in \mathbb{N}^B$$

of the visual words (quantized local visual features) that occur inside x_i^j (B represents the number of unique words in the visual codebook which, as usual, is learned from training images during an offline prior stage; more about this will be discussed later in "developmental learning"). The foreground content for the overall video is computed as the average of the content of the foreground regions in all its frames, i.e

$$H(z_i, x_i) = \frac{\sum_j h(z_i^j, x_i^j)}{|z_i|}$$

where $|z_i|$ is the number of frames in z_i .

For an object class k in a selected foreground of a frame, assume a multinomial gaussian distribution defined by parameters $\theta_k^F = \{\mu_k, \Sigma_k\}$ corresponding to this object class (k) in this foreground. For members of that k_{th} object class, the distributions $H(z_i, x_i)$ and

$h(z_i^j, x_i^j)$, i.e., the overall (average) foreground for the video, and the foregrounds in each of the constituent frames, are generated from a common model with parameters θ_k^F. Let label $l \in \{1,\ldots, K\}$ denote the unknown cluster label of video z_i, which we assume to be drawn from a multinomial mixture with mixture coefficients $\pi = \{\pi_1,\ldots,\pi_k\}$. Then for video z_i with label l_i, its foreground histogram $H(z_i, x_i)$ is drawn from the normal distribution θ_k^F with mean and covariance μ_l and Σ, so

$$H(z_i, x_i) \sim \mathcal{N}(\mu_l, \Sigma_l).$$

Reducing the number of parameters to be estimated we assume the covariance Σ_k of each cluster k to be diagonal: $\Sigma_k = \mathrm{diag}(\lambda_{k1},\ldots,\lambda_{kW})^{-1}$. Finally, each video is assumed to have its own independent background model (defined by parameters θ_k) which can be left unresolved for the current object-discovery objective. The figure below summarizes the above description of the generative model.

We maximize the likelihood of the model by marginalizing over the labels, which we treat as hidden variables. I.e., our objective is to find parameters $\theta = \{\theta^F, \pi\}$ and foreground regions $x = \{x_1,\ldots, x_n\}$ to maximize:

$$p(z|x,\theta)p(x) = \prod_{i=1}^{N} p(z_i|x_i,\theta)\,p(x_i)$$

which can be expanded to

$$= \prod_{i=1}^{N}\sum_{k=1}^{K} p(z_i, l_i = k|x_i,\theta)p(x_i)$$

This corresponds to a video extension of results showing a generative model for unsupervised object discovery in still images; the details appear in the accompanying article (Chandrashekar, Torresani & Granger, 2012). As in that work, we can maximize the proposed penalized likelihood via expectation maximization (EM), alternating between estimating the distribution over the cluster labels l_i for each video z_i, and solving for the foreground models and locations.

If we treat the frames of a video as still images, then the previous work (Chandrashekar et al., 2012) has shown how the foreground can be localized in them. We have subsequently derived an extended method whereby not just the appearance model, but also the video's optic-flow constraints, can limit the search for object foregrounds in video frames. Unlike the appearance model which is applied across videos in the dataset, the motion model is applied only within each video, ensuring that videos with very different motion signatures for the same objects can still contribute to the appearance model; i.e., the learned appearance model generalizes over different motion signatures.

Just as in our previous object recognition and localization methods, we can use two distinct methods for foregrounding: either rectangular bounding boxes, or bottom---up segmentation via superpixels.

7. Developmental learning

All of the methods described, as well as cited work from other labs that is referenced, contain a dependency on initialization, which can also be thought of as identifying and setting priors. For unsupervised learning, the initialization typically has a substantive impact on the quality of the final results. The parameters requiring initialization in our model are: mixture coefficients (π_k); histogram means (μ_k) and variances (Σ_k); and foreground masks for all video frames (x_i^j). Our first versions of these models initialized foreground masks by matching all pairs of images, performing a co-segmentation, which is an expensive process. Even this step was used only for stills; it was dispensed with in the case of videos, in favor of using motion---based segmentation to get initial estimates of foreground masks.

A new method has been developed to use large amounts of unlabeled data to automatically acquire approximations of priors. The method is generative and generates multiclass classifications (both as opposed to discriminative methods such as SVMs).

The intuition is that of a "developmental" stage in which the system uses a specialized set of rules on masses of otherwise uninterpreted data, to generate an initial tree that will correspond to a large vocabulary of features and collections of features. These then will be used from then on, in what an be thought of as the subsequent "adult" phase of the system, for the tasks we have been studying (recognition, classification, localization). The data structure acquired by the method is a hierarchy that contains information about vocabulary features and their relations to each other; it is termed a branching object-relation notation (BORN).

These trees are intuitively related to "vocabulary trees" of the kind described by Nister & Stewenius (2006), for instance: tree structures that capture a vocabulary of image features in a hierarchical form obtained by recursive application of K---means. We treat images as collections of objects, embedded in various settings. The objects *are represented in* the BORN structures which are constructed from very large collections of unlabeled images.

If the task is image retrieval alone, low-level feature-based representations may suffice, but for tasks of recognition and localization, richer representations will be beneficial. From a collection of unlabeled data (*U*) we identify object---like regions (via a published set of methods for images, and via those methods supplemented with optic flow information for videos). That collection of regions, O, is represented using simple bag-of-visual-words

(bovw) modeling, where $h(o)$ is the histogram of features for object region $o \in O$. We recursively cluster this collection (similar to Nister & Stewenius 2006) by applying a gaussian mixture model on the dataset to organize the appearance based clusters in the form of a tree.

At each node *t* of the BORN tree, we have a collection of object regions O^t. We assume a generative framework in which $h(o^t)$ is modeled as a random variable drawn from a gaussian distribution with parameters $\theta_l^t = \{\mu_l^t, \Sigma_l^t\}$ i.e., the histogram is related to a normal multinomial gaussian:

$$h(o^t) \sim \mathcal{N}(\mu_l^t, \Sigma_l^t)$$

where $l \in \{1,\ldots, K\}$ denotes the (unknown) cluster label of region *o* at node *t* in the tree. The label *l* is assumed to be drawn from a multinomial distribution with mixture parameters $\pi = \{\pi_1,\ldots,\pi_K\}$.

We again use the EM algorithm to maximize the likelihood of the model by marginalizing over the cluster labels; the likelihood function is:

$$p(O^t|\theta^t) = \prod_o \sum_{k=1}^{K} p(o, l_o = k|\theta)$$

For each cluster, a new child node is created in the BORN tree under node *t*. The cluster means and the variances $(\theta_{1..K}^n)$ for the child nodes are recorded and each cluster is further subdivided using the same process, continuing until a maximum allowed tree depth. The algorithm for producing the branching object-region notion tree is stated below:

Input: U - A set of unlabeled images.
Output: $BORN$, rooted at r
$Z = \emptyset$
for *image* $i \in U$ **do**
  O_i = Object_Region_Detector(i)
  $Z \leftarrow Z \cup O_i$
end
Init: $Z^r = Z$
Q = {r};
while $Q \neq \emptyset$ **do**
  t ← a node in Q
  $Q \leftarrow Q - t$
  if *SUBDIVIDE*$(Z^t) = true$ **then**
    $[\theta^t_{1..K}, C_{1..K}]$ = GMM (Z^t, K)
    foreach *Cluster* C_k **do**
      Node c
      $Z^c = C_k$
      $B^t_k = \theta_k$
      $Q \leftarrow Q \cup c$
    end
  end
end

3: Branching object-recognition tree

Once a BORN structure has been built from a large unlabeled dataset, we can use it to perform image retrieval. Every image in the retrieval dataset is encoded via the BORN tree. The similarity between a query image and an image in the database is determined by comparing the paths that are taken through the tree, by object regions from the images.

Each object region in an image is first described as a histogram (using bag-of-visual-words notation). Then the similarity between two images *q* and *d* can be computed:

$$s(q,d) = \left\| \frac{q}{\|q\|} - \frac{d}{\|d\|} \right\|$$

$$q_i = n_i w_i$$

$$d_i = m_i w_i$$

$$w_i = \ln \frac{N}{N_i}$$

where w_i is the weight of each node *i* in the tree. The variables n_i and m_i are the number of descriptor vectors of the query and the database image respectively, with a path through node *i* in the tree. *N* is the total number of images in the database, and N_i is the number of images that have at least one object region passing through node *i*.

Once a BORN tree has been constructed, subsequent learning, which corresponds to normal learning approaches such as learning label trees, can be thought of as "adult"

learning, using the results of the developmental creation of the BORN tree as setting priors for the adult stage (and thus substantially improving both learning rate and accuracy). Our method for object detection and classification, described earlier (and in Chadrashekar and Granger 2012), can be modified to use the BORN representation, constructing a label tree as a subgraph of the BORN tree.

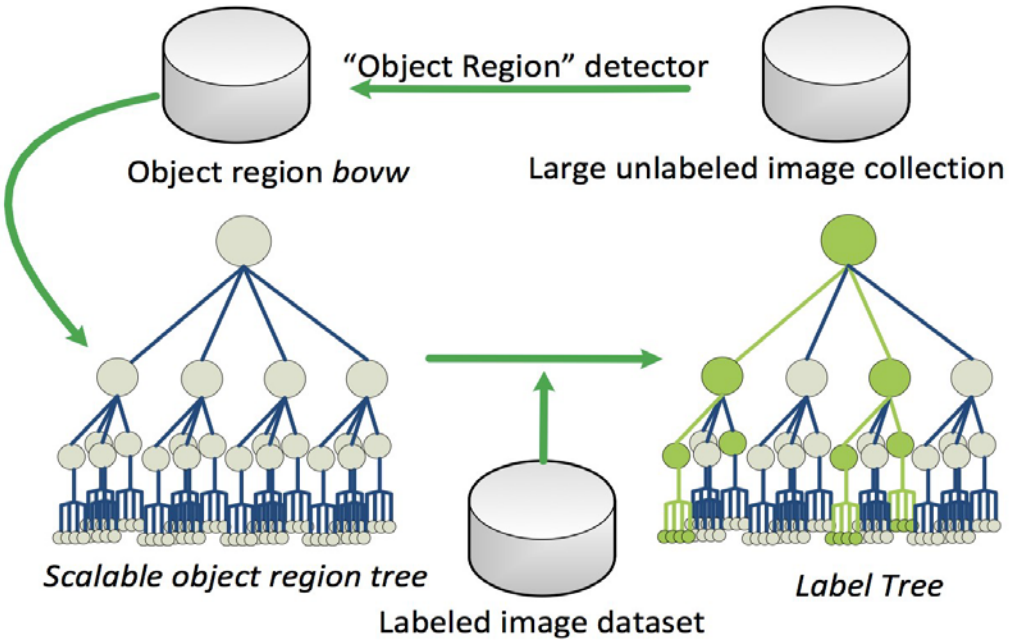
Classifiers that learn labeled trees have been shown to be more efficient than typical approaches that learn 1-versus-rest classifiers, as well as reducing recognition time — often to order log of the size of the learned labeled tree (Deng et al 2011; Bengio et al., 2010). All such methods, however, perform splits of the data by supervised means, learning hyperplanes. In contrast, we introduced methods using unsupervised clustering and localization via maximization of a single likelihood objective (Chandrashekar & Granger 2012; Chandrashekar et al., 2012). In particular, we constructed labeled trees via purely unsupervised splits of the data, iteratively "purifying" the clusters according to whether the supervised labels, within the unsupervised clusters, were consistent. Combining the two methods above can be seen as providing a natural framework for performing joint recognition and localization to construct hierarchical representations; moreover, the method is generative and thus can be used for multiclass classification.

The constructed labeled tree, as a subset of the BORN tree, contains only nodes where $N^t > 0$, i.e., all nodes in the BORN tree through which labeled training data has traversed. Thus the primary task for adult learning is at each node *t*, the labeled data is to be divided into *K* clusters. This is accomplished by performing a generative clustering and foreground localization task by maximizing the objective function specified in Chandrashekar, Torresani & Granger (2012). The clusters created are treated as child nodes of BORN tree nodes, with gaussian foreground parameters stored at the branch. We then examine each cluster to see if it needs to be further split, via an algorithm very similar to that described earlier for the CSL algorithm:

Input: $BORN$, Dataset: $Z \leftarrow \{< z_i, y_i >\}$ with labels $Y \leftarrow \{y_i \in \{1, 2, 3..K\}\}$
Output: Label Tree LT
Init: $t \leftarrow$ root node of $BORN$;
$Z^t \leftarrow Z$
$Q \leftarrow t$
while $Q \neq \emptyset$ **do**
  $t \leftarrow$ node in Q
  $Q \leftarrow Q - t$
  $Y^t \leftarrow Y(Z^t)$
  $N^t \leftarrow |Z^t|$
  if $|Y^t| > 1$ **then**
    Init: x^t, μ^t and Σ^t using node $t \in BORN$
    Compute θ^t, x^t by maximing $\mathcal{L}$
    for $i = 1 : |B^t|$ **do**
      $c \leftarrow B_i^t$
      if $C_i \neq \emptyset$ **then**
        $Z^c = C_i$
        $B_i^t = \theta_i$
        $Q \leftarrow Q \cup c$
      end
    end
  end
end

4: Constructed labeled tree algorithm

The process of building BORN representations developmentally, and using them for embedding labeled trees in subsequent adult learning, is illustrated here:



5: BORN representation

In sum, this has been an initial proof of principle effort to investigate new approaches to processing images, with an emphasis on moving images. Little prior work has been done on the processing of purely unlabeled images, let alone unlabeled video images, despite the fact

that the task of labeling data is among the most expensive (human-intensive) components of image processing. We studied the development of novel methods for visual processing of images and moving scenes. The work has substantially exceeded the initial proof of principle aims: we have developed novel algorithms for object recognition and localization in both still images and in videos, and have carried out initial evaluations comparing the new methods with previous approaches in the literature. The results have been highly promising and already have led to two publications (as well as a review paper). Ongoing work is aimed at extending the initial findings to include a suite of advanced approaches to the task of processing time-varying images.

8. Publications:

Chandrashekar A, Granger R (2012). Derivation of a novel efficient supervised learning algorithm from cortical-subcortical loops. Frontiers Comput Neurosci., 5: 50.
doi: 10.3389/fncom.2011.00050

Chandrashekar A, Torressani L, Granger R (2012). Learning what is where from unlabeled images: Joint localization and clustering of foreground objects. Dartmouth BELAB Tech Report 2012-3; submitted to Machine Learning Journal.

Granger R (2011). How brains are built: Principles of computational neuroscience. Cerebrum; The Dana Foundation. http://dana.org/news/cerebrum/detail.aspx?id=30356

9. References

A.Rodriguez, J.Whitson, R.Granger. Derivation and analysis of basic computational operations of thalamocortical circuits. J. Cognitive Neurosci., 16: 856--877, 2004.

R.Granger. Engines of the brain: The computational instruction set of human cognition. AI Magazine, 27: 15–32, 2006.

A. Felch and R. Granger. The hypergeometric connectivity hypothesis: Divergent performance of brain circuits with different synaptic connectivity distributions. Brain Research, (1202):3–13, 2008.

R. Granger. How brains are built: Principles of computational neuroscience. Cerebrum; The Dana Foundation, http://dana.org/news/cerebrum/detail.aspx?id=30356, 2011.

A. Chandrashekar, L. Torressani, and R. Granger. Learning what is where from unlabeled images: joint localization and clustering of foreground objects. BELAB Tech Rpt 3, 2012.

A. Chandrashekar and R. Granger. Derivation of a novel efficient supervised learning algorithm from cortical--subcortical loops. Frontiers in computational neuroscience., 5, 2012.

B.C. Russell, A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In CVPR, 2006.

M. Fritz and B. Schiele. Decomposition, discovery and detection of visual categories using topic models. In CVPR, 2008.

G.J.Brostow,J.Shotton,J.Fauqueur,andR.Cipolla.Segmentationandrecognitionusingstructurefrommotion point clouds. In ECCV, 2008.

L. Ladick, P. Sturgess, K. Alahari, C. Russell, and P Torr. What,where and how many? combining object detectors and crfs. In ECCV, 2010.

X.Bai, J.Wang, D.Simons, and G.Sapiro. Video snapcut: Robust video object cutout using localized classiMiers. In siggraph, 2009.

B. Price, B. Morse, and S. Cohen. Livecut: Learning--based interactive video segmentation by evaluation of multiple propagated cues. In iccv, 2009.

J. Malik and J. Shi. Motion segmentation and tracking using normalized cuts. In ICCV, 1998.

W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In ICCV, 2009.

T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In ECCV, 2010.

A.Vazquez--Reina, H.Avidan, H.Pfister, and E.Miller. Multiple hypothesis video segmentation from superpixel flows. In ECCV, 2010.

Y. Huang, Q. Liu, and D. Metaxas. Video object segmentation by hypergraph. In CVPR, 2009.

M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video

segmentation. In CVPR, 2010.

Y.J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In ICCV, 2011.

P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In VS-PETS, 2005.

I.. Laptev. On space-time interest points. Int. Journal of Computer Vision, 2005.

J.C.Niebles, H.Wang, and L.Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. International Journal of Computer Vision, 2008.

G. Willems, T. Tuytelaars, and L.J.V. Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In ECCV, 2008.

Jia Deng, Sanjeev Satheesh, Alex Berg, and Li Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In Proceedings of the Neural Information Processing Systems (NIPS), 2011.

S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In Proceedings of Neural Information Processing Systems (NIPS), 2010.

Appendix A

frontiers in
COMPUTATIONAL NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 10 January 2012
doi: 10.3389/fncom.2011.00050

Derivation of a novel efficient supervised learning algorithm from cortical-subcortical loops

Ashok Chandrashekar[1]* and Richard Granger[2]

[1] Department of Computer Science, Dartmouth College, Hanover, NH, USA
[2] Psychological and Brain Sciences, Thayer School of Engineering and Computer Science, Dartmouth College, Hanover, NH, USA

Edited by:
Hava T. Siegelmann, Rutgers University, USA

Reviewed by:
Thomas Boraud, Universite de Bordeaux, France
Taro Toyoizumi, RIKEN BSI, Japan

***Correspondence:**
Ashok Chandrashekar, Department of Computer Science, Dartmouth College, Hinman Box 6211, Hanover, NH 03775, USA.
e-mail: ashok.chandrashekar@dartmouth.edu

Although brain circuits presumably carry out powerful perceptual algorithms, few instances of derived biological methods have been found to compete favorably against algorithms that have been engineered for specific applications. We forward a novel analysis of a subset of functions of cortical–subcortical loops, which constitute more than 80% of the human brain, thus likely underlying a broad range of cognitive functions. We describe a family of operations performed by the derived method, including a non-standard method for supervised classification, which may underlie some forms of cortically dependent associative learning. The novel supervised classifier is compared against widely used algorithms for classification, including support vector machines (SVM) and k-nearest neighbor methods, achieving corresponding classification rates – at a fraction of the time and space costs. This represents an instance of a biologically derived algorithm comparing favorably against widely used machine learning methods on well-studied tasks.

Keywords: biological classifier, hierarchical, hybrid model, reinforcement, unsupervised

1. INTRODUCTION

Distinct brain circuit designs exhibit different functions in human (and other animal) brains. Particularly notable are studies of the basal ganglia (striatal complex), which have arrived at closely-related hypotheses, from independent laboratories, that the system carries out a form of reinforcement learning (Sutton and Barto, 1990; Schultz et al., 1997; Schultz, 2002; Daw, 2003; O'Doherty et al., 2003; Daw and Doya, 2006); despite ongoing differences in the particulars of these approaches, their overall findings are surprisingly concordant, corresponding to a still-rare instance of convergent hypotheses of the computations produced by a particular brain circuit. Models of thalamocortical circuitry have not yet converged to functional hypotheses that are as widely agreed-on, but several different approaches nonetheless hypothesize the ability of thalamocortical circuits to perform unsupervised learning, discovering structure in data (Lee and Mumford, 2003; Rodriguez et al., 2004; Granger, 2006; George and Hawkins, 2009). Yet thalamocortical and striatal systems do not typically act in isolation; they are tightly connected in cortico-striatal loops such that virtually each cortical area interacts with corresponding striatal regions (Kemp and Powell, 1971; Alexander and DeLong, 1985; McGeorge and Faull, 1988). The resulting cortico-striatal loops constitute more than 80% of human brain circuitry (Stephan et al., 1970, 1981; Stephan, 1972), suggesting that their operation provides the underpinnings of a very broad range of cognitive functions.

We forward a new hypothesis of the interaction between cortical and striatal circuits, carrying out a hybrid of unsupervised hierarchical learning and reinforcement, together achieving a cortico-striatal loop algorithm that performs a number of distinct operations of computational utility, including supervised and unsupervised classification, search, object and feature localization, and hierarchical memory organization. For purposes of the present paper we focus predominantly on the particular task of supervised learning.

Traditional supervised learning methods typically identify class boundaries by focusing primarily on the class labels, whereas unsupervised methods discover similarity structure occurring within a dataset; two distinct tasks with separate goals, typically carried out by distinct algorithmic approaches.

Widely used supervised classifiers such as support vector machines (Vapnik, 1995), supervised neural networks (Bishop, 1996), and decision trees (Breiman et al., 1984; Buntine, 1992), are so-called discriminative models, which learn separators between categories of sample data without learning the data itself, and without illuminating the similarity structure within the data set being classified.

The cortico-striatal loop (CSL) algorithm presented here is "generative," i.e., it is in the category of algorithms that models data occurring within each presented class, rather than seeking solely to identify differences between the classes (as would a "discriminative" method). Generative models are often taken as performing excessive work in cases where the only point is to distinguish among labeled classes (Ng and Jordan, 2002). The CSL method may thus be taken as carrying out more tasks than classification, which we indeed will see it does. Nonetheless, we observe the behavior of the algorithm in the task of classification, and compare it against discriminative classifiers such as support vectors, and find that even in this restricted (though very widely used) domain of application, the CSL method achieves comparable classification as discriminative models, and uses far less computational cost to do so, despite carrying out the additional work entailed in generative learning.

The approach combines the two distinct tasks of unsupervised classification and reinforcement, producing a novel method for yet

another task: that of supervised learning. The new method identifies supervised class boundaries, as a byproduct of uncovering structure in the input space that is independent of the supervised labels. It performs solely unsupervised splits of the data into similarity-based clusters. The constituents of each subcluster are checked to see whether or not they all belong to the same intended supervised category. If not, the algorithm makes another unsupervised split of the cluster into subclusters, iteratively deepening the class tree. The process repeats until all clusters contain only (or largely) members of a single supervised class. The result is the construction of a hierarchy of mostly mixed classes, with the leaves of the tree being "pure" categories, i.e., those whose members contain only (or mostly) a single shared supervised class label.

Some key characteristics of the method are worth noting.

- Only unsupervised splits are performed, so clusters always contain only members that are similar to each other.
- In the case of similar-looking data that belong to distinct supervised categories (e.g., similar-looking terrains, one leading to danger and one to safety), these data will constitute a difficult discrimination; i.e., they will reside near the boundary that partitions the space into supervised classes.
- In cases of similar data with different class labels, i.e., difficult discriminations, the method will likely perform a succession of unsupervised splits before happening on one that splits the dangerous terrains into a separate category from the safe ones.

In other words, the method will expend more effort in cases of difficult discriminations. (This characteristic is reminiscent of the mechanism of support vectors, which identify those vectors near the intended partition boundary, attempting to place the boundary so as to maximize the distance from those vectors to the boundary.) Moreover, in contrast to supervised methods that provide expensive, detailed error feedback at each training step (instructing the method as to which supervised category the input should have been placed in), the present method uses feedback that is comparatively far more inexpensive, consisting of a single bit at each training step, telling the method whether or not an unsupervised cluster is yet "pure"; if so, the method stops for that node; if not, the method performs further unsupervised splits.

This deceptively simple mechanism not only produces a supervised classifier, but also uncovers the similarity structure embedded in the dataset, which competing supervised methods do not. Despite the fact that competing algorithms (such as SVM and Knn) were designed expressly to obtain maximum accuracy at supervised classification, we present findings indicating that even on this task, the CSL algorithm achieves comparable accuracy, while requiring significantly less computational resource cost.

In sum, the CSL algorithm, derived from the interaction of cortico-striatal loops, performs an unorthodox method that rivals the best standard methods in classification efficacy, yet does so in a fraction of the time and space required by competing methods.

2. CORTICO-STRIATAL LOOPS

The basal ganglia (striatal complex), present in reptiles as well as in mammals, is thought to carry out some form of reinforcement learning, a hypothesis shared across a number of laboratories (Sutton and Barto, 1990; Schultz et al., 1997; Schultz, 2002; Daw, 2003; O'Doherty et al., 2003; Daw and Doya, 2006). The actual neural mechanisms proposed involve action selection through a maximization of the corresponding reward estimate for the action on the task (see Brown et al., 1999; Gurney et al., 2001; Daw and Doya, 2006; Leblois et al., 2006; Houk et al., 2007 for a range of views on action selection). This reward estimation occurs in most models of the striatum through the regulation of the output of the neurotransmitter dopamine. Therefore, in computational terms we can characterize the functionality of the striatum as an abstract search through the space of possible actions, guided by dopaminergic feedback.

The neocortex and thalamocortical loops are thought to hierarchically organize complex fact and event information, a hypothesis shared by multiple researchers (Lee and Mumford, 2003; Rodriguez et al., 2004; Granger, 2006; George and Hawkins, 2009). For instance, in Rodriguez et al. (2004) the anatomically recognized "core" and "matrix" subcircuits are hypothesized to carry out forms of unsupervised hierarchical categorization of static and time-varying signals; and in Lee and Mumford (2003), George and Hawkins (2009), Riesenhuber and Poggio (1999), and Ullman (2006) and many others, hypotheses are forwarded of how cortical circuits may construct computational hierarchies; these studies from different labs propose related hypotheses of thalamocortical circuits performing hierarchical categorization.

It is widely accepted that these two primary telencephalic structures, cortex and striatum, do not act in isolation in the brain; they work in tight coordination with each other (Kemp and Powell, 1971; Alexander and DeLong, 1985; McGeorge and Faull, 1988). The ubiquity of this repeated architecture (Stephan et al., 1970, 1981; Stephan, 1972) suggests that cortico-striatal circuitry underlies a very broad range of cognitive functions. In particular, it is of interest to determine how semantic cortical information could provide top-down constraints on otherwise too-broad search during (striatal) reinforcement learning (Granger, 2011). In the present paper we study this interaction in terms of subsets of the leading extant computational hypotheses of the two components: thalamocortical circuits for unsupervised learning and the basal ganglia/striatal complex for reinforcement of matches and mismatches. If these bottom-up analyses of cortical and striatal function are taken seriously, it is of interest to study what mechanisms may emerge from the interaction of the two mechanisms when engaged in (anatomically prevalent) cortico-striatal loops.. We adopt straightforward and tractable simplifications of these models, to study the operations that arise when the two are interacting. **Figure 1** illustrates a hypothesis of the functional interaction between unsupervised hierarchical clustering (uhc; cortex) and match-mismatch reinforcement (mm; striatal complex), constituting the integrated mechanism proposed here.

The interactions in the simplified algorithm are modeled in part on mechanisms outlined in Granger (2006): a simplified model of thalamocortical circuits produces unsupervised clusters of the input data; then, in the CSL model, the result of the clustering, along with the corresponding supervised labels, are examined by a simplified model of the striatal complex. The full computational models of the thalamocortical hierarchical clustering and sequencing circuit and striatal reinforcement-learning circuit yield interactions that are under ongoing study, and will, it is hoped, lead to further derivation of additional algorithms. For

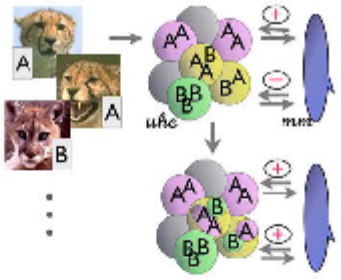


FIGURE 1 | Simplified hypothesis of the computations of cortico-striatal interaction. (Left) Inputs (e.g., images) arrive together with associated labels (A, B, etc.). (Top right) Unsupervised hierarchical clustering (uhc) categorizes the inputs by their similarity, without initial regard to labels. (Far right) The match–mismatch mechanism (mm) matches the class membership labels within each of these unsupervised categories. If the members of a category all have the same (matching) labels, a "+" is returned; if there are mismatches, a "−" is returned. (Bottom right) In the latter case, the clustering mechanism iteratively deepens the hierarchical tree by creating subclusters (e.g., central node in the diagram); these in turn are checked for label consistency, as before. The process iterates until the leaf nodes of the unsupervised tree contain only category members of a single label. (See text).

the present paper, we use just small subsets of the hypothesized functions of these structures: solely the hypothesized hierarchical clustering function of the thalamocortical circuit, and a very-reduced subset of the reinforcement-learning capabilities of the striatal complex, such that it does nothing more than compare (match / mismatch) the contents of a proposed category, and return a single bit corresponding to whether the contents all have been labeled as "matching" each other (1) or not (0). This very-reduced RL mechanism can be thought of simply as rewarding or punishing a category based on its constituents. In particular the proposed simplified striatal mechanism returns a single bit (correct/incorrect) denoting whether the members of a given unsupervised cluster all correspond to the same supervised "label." If not, the system returns a "no" ("−") to the unsupervised clustering mechanism, which in turn iterates over the cluster producing another, still unsupervised, set of subclusters of the "impure" cluster. The process continues until each unsupervised subcluster contains members only (or mostly, in a variant of the algorithm) of a single category label.

In sum, the mechanism uses only unsupervised categorization operations, together with category membership tests. These two mechanisms result in the eventual iterative arrival at categories whose members can be considered in terms of supervised classes.

Since only unsupervised splits are performed, categories (clusters) always contain only members that are similar to each other. The tree may generate multiple terminal leaves corresponding to a given class label; in such cases, the distinct leaves correspond to dissimilar class subcategories, eventually partitioned into distinct leaf nodes. The mechanism can halt rapidly if all supervised classes correspond to similarity-based clusters; i.e., if class labels are readily predictable from their appearance. This corresponds to an "easy" discrimination task. When this is not the case, i.e., in instances where similar-looking data belong to different labeled categories (e.g., similar mushrooms, some edible and some poisonous), the mechanism will be triggered to successively subdivide clusters into subclusters, as though searching for the characteristics that effectively separate the members of different labels.

In other words, less work is done for "easy" discriminations; and only when there are difficult discriminations will the mechanism perform additional steps. The tree becomes intrinsically unbalanced as a function of the lumpiness of the data: branches of the tree are only deepened in regions of the space where the discriminations are difficult, i.e., where members of two or more distinct supervised categories are close to each other in the input space. This property is reminiscent of support vectors, which identify boundaries in the region where two categories are closest (and thus where the most difficult discriminations occur).

A final salient feature of the mechanism is its cost. In contrast to supervised methods, which provide detailed, expensive, error feedback at each training step (telling the system not only when a misclassification has been made but also exactly which class should have occurred), the present method uses feedback that by comparison is extremely inexpensive, consisting of a single bit, corresponding to either "pure" or "impure" clusters. For pure clusters, the method halts; for impure clusters, the mechanism proceeds to deepen the hierarchical tree.

As mentioned, the method is generative, and arrives at rich models of the learned input data. It also produces multiclass partitioning as a natural consequence of its operation, unlike discriminative supervised methods which are inherently binary, requiring extra mechanisms to operate on multiple classes.

Overall, this deceptively simple mechanism not only produces a supervised classifier, but also uncovers the similarity structure embedded in the dataset, which competing supervised methods do not. The terminal leaves of the tree provide final class information, whereas the internal nodes provide further information: they are mixed categories corresponding to meta labels (e.g., superordinate categories; these also can provide information about which classes are likely to become confused with one another during testing.

In the next section we provide an algorithm that retains functional equivalence with the biological model for supervised learning described above while abstracting out the implementation details of the thalamocortical and striatal circuitry. Simplifying the implementation enables investigation of the algorithmic properties of the model independent of its implementation details (Marr, 1980). It also, importantly, allows us to test our model on real-world data and compare directly against standard machine learning methods. Using actual thalamocortical circuitry to perform the unsupervised data clustering and the mechanism for the basal ganglia to provide reinforcement feedback, would be an interesting task for the distinct goal of investigating potential implementation-level predictions; this holds substantial potential for future research.

We emphasize that our focus is to use existing hypotheses of telencephalic component function already posited in the literature; these mechanisms lead us to specifically propose a novel method by which supervised learning is achieved by the unlikely route of

combining unsupervised learning with reinforcement. This kind of computational-level abstraction and analysis of biological entities continues in the tradition of many prior works, including Suri and Schultz (2001), Schultz (2002), Daw and Doya (2006), Lee and Mumford (2003), Rodriguez et al. (2004), George and Hawkins (2009), Marr (1980), Riesenhuber and Poggio (1999), Ullman (2006), and many others.

3. SIMPLIFIED ALGORITHM

In our simplified algorithm, we refer to a method which we term *PARTITION*, corresponding to any of a family of clustering methods, intended to capture the clustering functionality of thalamocortical loops as described in the previous sections; and we refer to a method we term *SUBDIVIDE*, corresponding to any of a family of simple reinforcement methods, intended to capture the reinforcement-learning functionality of the basal ganglia/striatal complex as described in the previous sections. These operate together in an iterative loop corresponding to cortico-striatial (cluster–reinforcement) interaction: *SUBDIVIDE* checks for the "terminating" conditions of the iterative loop by examining the labels of the constituents of a given cluster and returning a *true* or *false* response. The resulting training method builds a tree of categories which, as will be seen, has the effect of performing supervised learning of the classes. The leaves of the tree contain class labels; the intermediate nodes may contain members of classes with different labels. During testing, the tree is traversed to obtain the label prediction for the new samples. Each data sample (belonging to one of K labeled classes) is represented as a vector $x \in R^m$. During training, each such vector x_i has a corresponding label $y_i \in 1, \ldots K$. (The subsequent "Experiments" section below describes the methods used to transform raw data such as natural images into vector representations in a domain-dependent fashion.)

3.1. TRAINING

The input to the training procedure is the training dataset consisting of (x_i, y_i) pairs where x_i is an input vector and y_i is its intended class label, as in all supervised learning methods. The output is a tree that is built by performing a succession of unsupervised splits of the data. The data corresponding to any given node in the tree is a subset of the original training dataset with the full dataset corresponding to the root of the tree. The action performed with the data at a node in the tree is an unsupervised split, thereby generating similarity-based clusters (subclusters) of the data within that tree node. The unsupervised split results in expansion (deepening) of the tree at that node, with the child nodes corresponding to the newly created unsupervised data clusters. The cluster representations corresponding to the children are recorded in the current node. These representations are used to determine the local branch that will be taken from this node during testing, in order to obtain a class prediction on a new sample. For each of the new children nodes, the labels of the samples within the cluster are examined, and if they are deemed to be sufficiently pure, i.e., a sufficient percentage of the data belong to the same class, then the child node becomes a (terminal) leaf in the tree. If not, the node is added to a queue which will be subjected to further processing, growing the tree. This queue is initialized with the root of the tree. The procedure (sketched in **Algorithm 1** below) proceeds until the queue becomes empty.

To summarize the mechanism, the algorithm attempts to find clusters based on appearance similarity, and when these clusters don't match with the intended (supervised) categories, reinforcement simply gives the algorithm the binary command to either split or not split the errant cluster. The behavior of the algorithm on sample data is illustrated in **Figure 2**. The input space of images is partitioned by successively splitting the corresponding training samples into subclusters at each step.

3.1.1. *Picking the right branch factor*

Since the main free parameter in the algorithm is the number of unsupervised clusters to be spawned from any given node in the hierarchy, the impact of that parameter on the performance of the algorithm should be studied. This quantity corresponds to the branching factor for the class tree. We initially propose a single parameter as an upper bound for the branch factor: K^{max}, which fixes the largest number of branches that can be spawned from any node in the tree. Through experimentation (discussed in the Results section) we have determined that (i) very small values for this parameter result in slightly lower prediction accuracy;

```
Input: Dataset: X = {x_i ⊂ R^M} with labels
       Y = {y_i ⊂ [1, 2, 3..K]}
Output: Class Tree: A tree rooted at the node
        TRoot
Init: TRoot.X = X, TRoot.Y = Y; TRoot.Labels =
LABELSET(Y)
Q = []; Add(Q, TRoot);
while Q is not empty do
    qn = First node in Q
    if SUBDIVIDE(X_qn, Y_qn) = true then
        [Centroids, Clusters] = PARTITION (X_qn,
        K*)
        foreach Cluster C_k do
            Node T
            T.X = Clusters[k]
            T.Labels = LABELSET(Y(T.X))
            qn.Branches[k] = Centroids[k]
            qn.Children[k] = T
            Add(Q, T)
        end
    end
end
```

Algorithm 1 | A sketch of the CSL learning algorithm. The method constructs a meta class tree, which records unsupervised structure within the data, as well as providing a means to perform class prediction on novel samples. The function termed *PARTITION* denotes an unsupervised clustering algorithm, which can in principle be any of a family of clustering routines; selections for this algorithm are described later in the text. The subroutine *SUBDIVIDE* determines if the data at a tree node qn all belong to the same class or not. If the data come from multiple classes, *SUBDIVIDE* returns *true* and otherwise, *false*. See text for further description of the derivation of the algorithm.

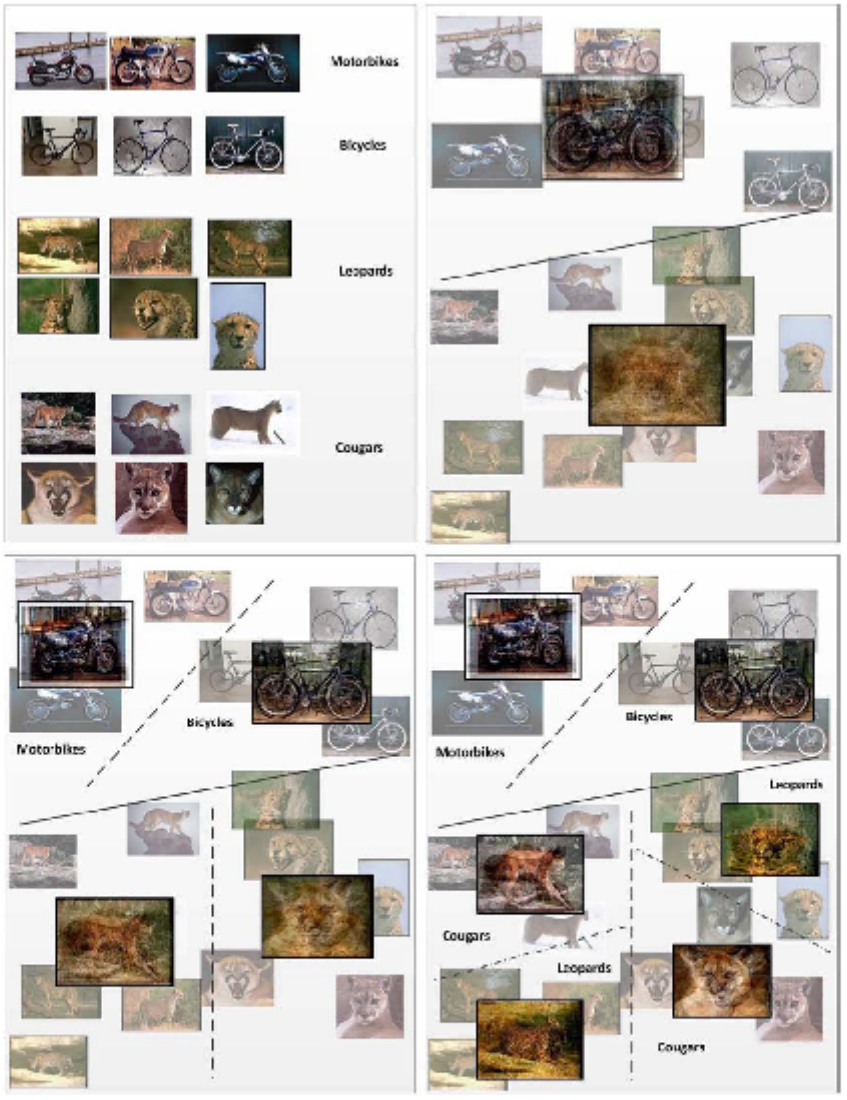


FIGURE 2 | A simplified illustration of the iterative learning process with labeled data. Images are successively split into two partitions in an unsupervised fashion (i.e., by similarity). The partitioning of data proceeds iteratively until the clusters formed are pure with respect to their labels. For each split, on either side of the dividing hyperplane, the means are shown as an overlay of the images that fall on the corresponding side of the hyperplane.

(II) for sufficiently large values, the parameter setting has no significant impact on the performance efficacy of the classifier; and (III) larger values of the parameter modestly increase the memory requirements of the clustering algorithm and thus the runtime of the learning stage (see the Results section below for further detail). (It is worth noting that selection of the best branch factor value

may be obtained by examination of the distribution of the data to be partitioned in the input space, enabling automatic selection of the ideal number of unsupervised clusters without reference to the number of distinct labeled classes that occur in the space. Future work may entail the study of existing methods for this approach, Baron and Cover, 1991; Teh et al., 2004, as potential adjunct improvements to the CSL method.)

3.2. TREE PRUNING

Categorization algorithms are often subject to overfitting the data. Aspects of the CSL algorithm can be formally compared to those of decision trees, which are subject to overfitting.

Unlike decision trees, the classes represented at the leaves of the CSL tree need not be regarded as conjunctions of attribute values on the path from the root, and can be treated as fully represented classes by themselves. (We refer to this as the "leaf independence" property of the tree; this property will be used when we describe testing of the algorithm in the next section.) Also, since the splits are unsupervised and based on multidimensional similarity (also unlike decision trees), they exhibit robustness w.r.t. variances in small subsets of features within a class.

Both of these characteristics (leaf independence and unsupervised splitting) theoretically lead to predictions of less overfitting of the method.

In addition to these formal observations, we studied overfitting in the CSL method empirically. Analogously to decision trees, we could choose either to stop growing the tree before all leaves were perfectly pure (and potentially overfit), or to build a full tree and then somewhat prune it back. Both methods improve the overfitting problem observed in decision trees. Experiments with both methods in the CSL algorithm found that neither one had a significant effect on prediction accuracy. Thus, surprisingly, both theoretical and empirical studies find that the CSL class trees generalize well without overfitting; the method is unexpectedly resistant to overfitting.

3.3. TESTING

During testing, the algorithm is presented with previously unseen data samples whose class we wish to predict. The training phase created an appearance-based class hierarchy. Since the tree, including the "pure class" leaves, is generative in nature, there are two alternative procedures for class prediction. One is that of descending the tree, as is done in decision trees. However, in addition, the "leaf independence" property of the CSL tree, as described in the previous section (which does not hold for decision trees), enables another testing method, which we refer to as KNN-on-leaves, in which we only attend to the leaf nodes of the tree, as described in the second subsection below. (This property does not hold for decision trees, and thus this additional testing method cannot be applied to decision trees). The two test methods have somewhat different memory and computation costs and slightly different prediction accuracies.

3.3.1. *Tree descent*

This approach starts at the root of the class tree, and descends. At every node, the test datum is compared to the cluster centroids stored at the node to determine the branch to take. The branch taken corresponds to the closest centroid to the test datum; i.e., a decision is made locally at the node. This provides us a unique path from the root of the class hierarchy to a single leaf; the stored category label at that leaf is used to predict the label of the input. Due to tree pruning (described above), the leaves may not be completely pure. As a result, instead of relying on any given class being present in the leaves, the posterior probabilities for all the categories represented at the leaf are used to predict the class label for the sample.

3.3.2. *KNN-on-leaves*

In this approach, we make a note of all the leaves in the tree, along with the cluster representation in the parent of the leaf node corresponding to the branch which leads to the leaf. We then do K-nearest neighbor matching of the test sample with all these cluster centroids that correspond to the leaves. The final label predicted corresponds to the label of the leaf with the closest centroid. This approach implies that only the leaves of the tree need to be stored, resulting in a significant reduction in the memory required to store the learned model. However, a penalty is paid in recognition time, which in this case is proportional to the number of leaves in the tree.

The memory required to store the model in the tree descent approach is higher than that for the KNN-on-leaves approach. However, tree descent offers a substantial speedup in recognition, as comparisons need to be performed only along a single path through the tree from the root to the final leaf. The algorithm is sketched below in **Algorithm 2**.

We expect that the KNN-on-leaves variant will yield better prediction accuracy as the decision is made at the end of the tree and

```
Input: x ∈ R^M , Class tree: TRoot
Output: y ∈ 1, 2, ...K
Init: Tree Node T = TRoot
while T is not leaf do
    mostSim = 0
    for k = 1: |T.Children| do
        sim = SIMILARITY(x, T.Centroids[k])
        if sim > mostSim then
            mostSim = sim
            branch = k
        end
    end
    T = T.Children[branch]
end
y = T.LabelSet
```

Algorithm 2 | A sketch of the tree descent algorithm for classifying a new data sample. The method starts at the root node and descends, testing the sample datum against each node encountered to determine the branch to select for further descent. The result is a unique path from the root to a single leaf; the stored category at that leaf is the prediction of the label of the input. In the event of impure leaves, the posterior probabilities for all categories in the leaf are used to predict the class label of the sample. See text for further description.

hence the partitioning of the input space is expected to exhibit better generalization. In the case of tree descent, since decisions are made locally within the tree, if the dataset has high variance, then it is possible that a wrong branch will be taken early on in the tree, leading to inaccurate prediction. This problem is common to a large family of algorithms, including decision trees. We have performed experiments to compare the two test methods: the results confirm that the KNN-on-leaves method exhibits marginally better prediction than the tree-descent method. The behavior of the two methods is illustrated in **Figure 3**.

4. CLUSTERING METHODS

The only remaining design choice is which unsupervised clustering algorithm to employ for successively partitioning the data during training, and the corresponding similarity measure. The choice can change depending on the type of data to be classified, while the overall framework remains the same, yielding a potential family of closely related variants of the CSL algorithm. This enables flexibility in selecting a particular unsupervised clustering algorithm for a given domain and dataset, without modifying anything else in the algorithm. (Using different clustering algorithms within the same class tree is also feasible as all decisions are made locally in the tree.)

There are numerous clustering algorithms from the simple and efficient k-means (Lloyd, 1982), self organizing maps (SOM; Kaski, 1997) and competitive networks (Kosko, 1991), to the more elaborate and expensive probabilistic generative algorithms like mixture of Gaussians, Probabilistic latent semantic analysis (PLSA; Hoffman, 1999) and Latent Dirichlet Allocation (LDA; Blei et al., 2003); each has merits and costs. Given the biological derivation of the system, we began by choosing k-means, a simple and inexpensive clustering method that has been discussed previously as a candidate system for biological clustering (Darken and Moody, 1990); the method could instead use SOM or competitive learning, two highly related systems. (It remains quite possible that more robust (and expensive) algorithms such as PLSA and LDA could provide improved prediction accuracy. Improvements might also arise by treating the data at every node as a mixture of Gaussians, and estimating the mixture parameters using the expectation maximization (EM) algorithm.)

4.1. *k*-MEANS

k-Means is one of the most popular algorithms to cluster n vectors based on distance measure into k partitions, where $k < n$. It attempts to find the centers of natural clusters in the data. The objective that *k*-means tries to minimize is the total *intra cluster*

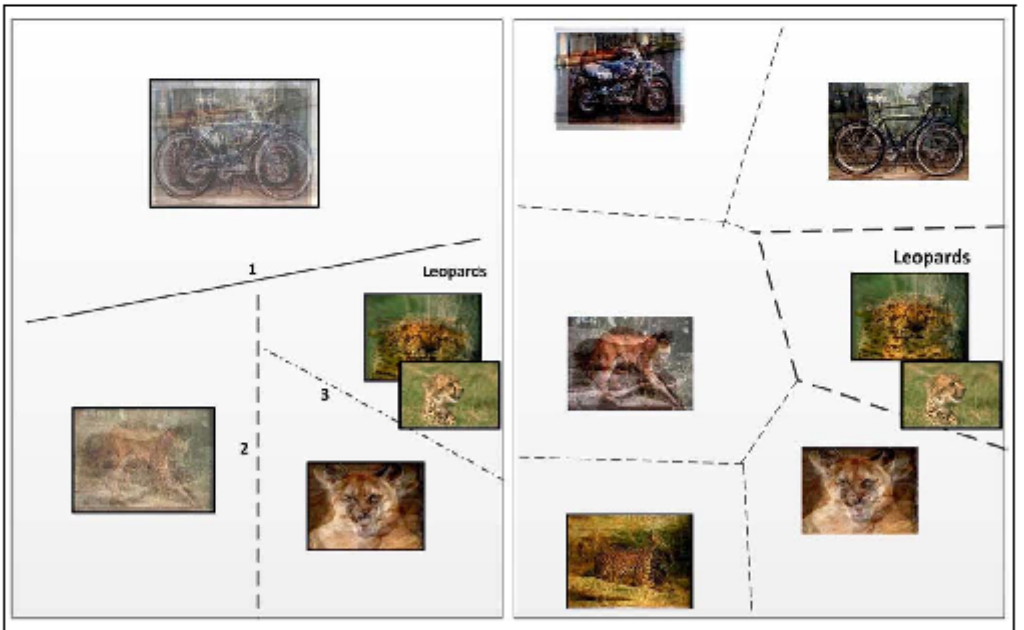


FIGURE 3 | Two methods by which the CSL algorithm predicts category membership at test time. (Left) Class prediction via hierarchical descent. At each step, a new (unknown) sample will fall on one or the other side of a classification hyperplane. The decision provides a path through the class tree at each node. At the leaves, the class prediction is obtained. The numbering gives the order in which the hyperplanes are probed. (Right) Class prediction using only leaves of class tree. All leaves are considered simultaneously; the test sample is compared to each leaf and the class prediction is obtained using KNN.

variance, or, the squared error function:

$$\Phi = \sum_{i=1}^{K} \sum_{x_j \in C_i} (x_j - \mu_i)^2$$

where there are K clusters S_i, $i = 1, 2, \ldots, K$ and μ_i is the centroid or mean point of all the points $x_j \in C_i$.

When k-means is used for the unsupervised appearance-based clustering at the nodes of the class tree, the actual means obtained are stored at each node, and the similarity measure is inversely proportional to the Euclidean distance.

4.1.1. Initializing clusters

In general, unsupervised methods are sensitive to initialization. We initialize the clustering algorithm at every node in the class tree as follows.

If we are at node l, with samples having one of K_l labels, we first determine the class averages of the K_l categories. (For every class, we remove the samples which are at least 2 standard deviations away from the mean of the class for the initialization These samples are considered for the subsequent unsupervised clustering.) If the number of clusters (branches), $K^* = min\ (K_l, K^{max})$ turns out to be equal to K_l, then the averages are used as the seeds for the clustering algorithm. If however $K^* < K_l$, then we use a simple and efficient method for obtaining the initial clusters by using an initial run of k-means on the K_l averages in order to obtain the K^* initial centroids. The data samples are assigned to the clusters using nearest neighbor mapping, and the averages of these K^* clusters are used as seeds for a subsequent run of the unsupervised clustering algorithm. (In our empirical experiments we have used the k-means++ variant of the popular clustering algorithm to obtain the initial cluster seeds; Arthur and Vassilvitskii, 2007.) **Figure 4** illustrates the initialization method. (While the method works relatively well, further studies indicate that other methods, which directly utilize the semantic structure of the labeled dataset, can result in even better performance. These alternate approaches are not discussed in this paper in order to keep the focus on introducing the core algorithm.) It is worth noting that the initialization method can be thought of in terms of a logically prior "developmental" period, in which no data is actually stored, but instead sampling of the environment is used to set parameters of the method; those parameters, once fixed, are then used in the subsequent performance of the then-"adult" algorithm (Felch and Granger, 2008).

5. EXPERIMENTS

The proposed algorithm performs a number of operations on its input, including the unsupervised discovery of structure in the data. However, since the method, despite being composed only of unsupervised clustering and reinforcement learning, can nonetheless perform supervised learning, we have run tests that involve using the CSL method solely as a supervised classifier. In addition to these tests of supervised learning alone, we then briefly describe some additional findings illustrating the CSL algorithm's power at tasks beyond the classification task (including the tasks of identifying structure in data, and localizing objects within images).

When viewed solely as a supervised classifier, the CSL method bears resemblances to two well-studied methods in machine learning and statistics, and we rigorously compare these. We compared

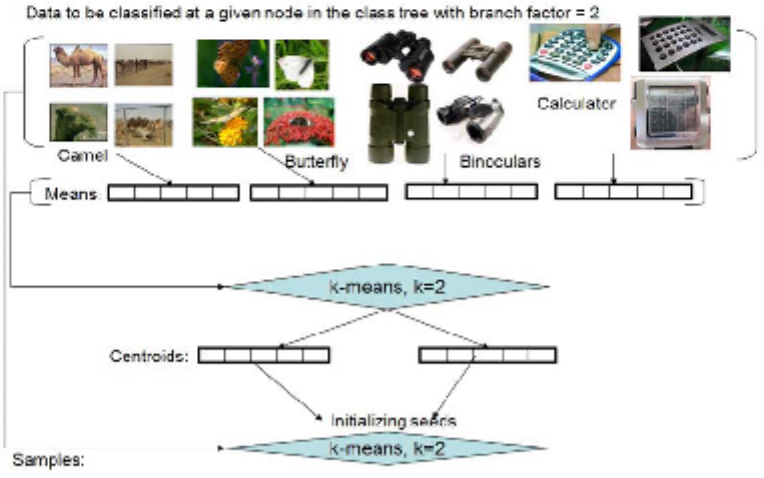


FIGURE 4 | Initialization of the unsupervised partitioning for a set of labeled training examples. The illustration depicts the process when working with image datasets. The method needs to be applied only when the desired number of clusters (in this example, 2) is less than the actual number of labeled classes represented in the dataset (in this example, 4) to be clustered.

the accuracy, and the time and space costs, of the CSL algorithm as a supervised classifier, against the support vector machine (SVM) and k-nearest neighbor (KNN) algorithms. Performance was examined on two well-studied public datasets.

For SVM, we have used the popular LibSVM implementation that is publicly available (Chang and Lin, 2001). This package implements the "one vs one" flavor of multiclass classification, rather than "one vs rest" variant based on the findings reported in Hsu and Lin (2002). After experimenting with a few kernels, we chose the linear kernel since it was the most efficient and especially since it provided the best SVM results for the high-dimensional datasets we tested. It is known that for the linear kernel a weight vector can be computed and hence the support vectors need not be kept in memory, resulting in low memory requirements and fast recognition time. However, this is not true for non-linear kernels where support vectors need to be kept in memory to get the class predictions at run time. Since we wish to compare the classifiers in the general setting and it is likely that the kernel trick may need to be employed to separate non-linear input space, we have retained the implementation of LibSVM as it is (where the support vectors are retained in memory and used during testing to get class prediction). We realize this may not be the fairest comparison for the current set of experiments, however, we believe that this setting is more reflective of the typical use case scenario where the algorithms will be employed.

For KNN we have hand coded the implementation and set the parameter $K = 1$ for maximum efficiency. (For the CSL algorithm with KNN-on-leaves, we use $K = 1$ as well.) The test bed is a machine running windows XP 64 with 8GB memory. We have not used hardware acceleration for any of the algorithms to keep the comparison fair.

We have used two popular datasets from different domains with very different characteristics (including dimensionality of the data) to fully explore the strengths and weaknesses of the algorithm. One is a subset of the Caltech-256 image set, and the other is a very high-dimensional dataset of neuroimaging data from fMRI experiments, that has been widely studied.

For both experiments, we performed multiple runs, differently splitting the samples from each class into training and testing sets (roughly equal in number). The results shown indicate the means and standard deviations of all runs.

5.1. OBJECT RECOGNITION

Our first experiment tests the algorithm for object recognition in natural still image datasets. The task is to predict the label for an image, having learned the various classes of objects in images through a training phase. We report empirical findings for prediction accuracy and computational resources required.

5.1.1. *Dataset*

The dataset used consists of a subset of the Caltech-256 dataset (Griffin et al., 2007) using 39 categories, each with roughly 100 instances. The categories were specifically chosen to exhibit very high between-category similarity, intentionally selected as a very challenging task, with high potential confusion among classes. The categories are:

- Mammals: bear, chimp, dog, elephant, goat, gorilla, kangaroo, leopard, raccoon, zebra
- Winged: duck, goose, hummingbird, ostrich, owl, penguin, swan, bat, cormorant, butterfly
- Crawlers (reptiles/insects/arthropods/amphibians): iguana, cockroach, grasshopper, housefly, praying mantis, scorpion, snail, spider, toad
- Inanimate objects: backpack, baseball glove, binoculars, bulldozer, chandeliers, computer monitor, grand piano, ipod, laptop, microwave.

We have chosen an extremely simple (and very standard) method for representing images in order to maintain focus on the description of the proposed classifier. First a feature vocabulary consisting of SIFT features (Lowe, 2004) is constructed by running k-means on a random set of images containing examples from all classes of interest; each image is then represented as a histogram of these features. The positions of the features and their geometry is ignored, simplifying the process and reducing computational costs. Thus each image is a vector $x \in R^m$, where m is the size of the acquired vocabulary. Each dimension of the vector is a count of the number of times the particular feature occurred in the image. This representation, known as the "Bag of Words," has been successfully applied before in several domains including object recognition in images (Sivic and Zisserman, 2003).

We ran a total of 8 trials, corresponding to 8 different random partitionings of the Caltech-256 data into training and testing sets. In each trial, we ran the test for each of a range of K_{max} values, to test this free parameter of the CSL model.

5.1.2. *Prediction accuracy*

The graph in top left of **Figure 5** compares the classifier prediction accuracy of the proposed algorithm with that of SVMs on the 39 subsets of Caltech-256 described earlier. As expected, the simplistic image representation scheme, and the readily confused category members, renders the task extremely difficult. It will be seen that all classifiers perform at a very modest success rate with this data, indicating the difficulty of the dataset and the considerable room for potential improvement in classification techniques.

The two variants of the CSL algorithm are competitive with SVM: SVM has an average accuracy of 23.9%; CSL with tree descent has an average accuracy of 19.4%; and CSL with KNN-on-leaves has an average prediction accuracy of 21.3%. The KNN algorithm alone performs relatively poorly, with an average prediction accuracy of 13.6%. Chance probability of correctly predicting a class is 1 out of 39 (2.56%).

It can be seen that the branch factor does not have a significant impact on error rates. This is possibly because the class tree grows until the leaves are pure, and the resulting internal structure, though different across choices of K^{max}, does not significantly impact the ultimate classifier performance as the hierarchy adapts its shape. Different internal structure could significantly affect the performance of the algorithm on tasks that depended on the similarity structure of the data, but for the sole task of supervised classification, the tree's internal nodes have little effect on prediction accuracy.

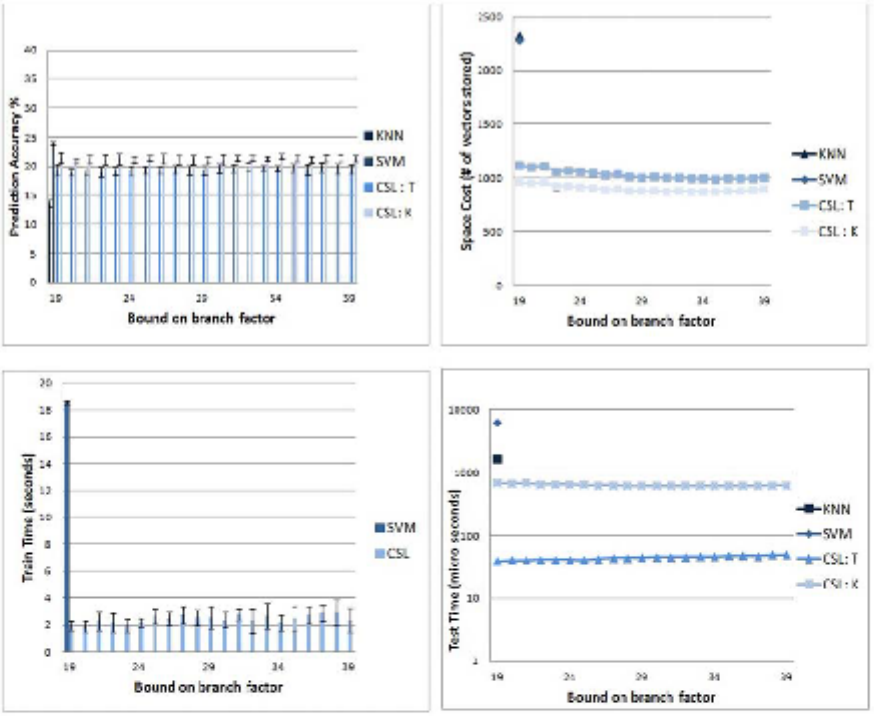


FIGURE 5 | Comparison of CSL and other standard classifiers. (Top left) Prediction accuracy of the CSL classifier on the Caltech-256 subsets. The scores in blue are the rates achieved by the CSL classifier. Scores in pink are from standard multiclass SVM (see text). (Top right) Memory requirements for CSL with respect to the branch factor parameter. The figure shows that the parameter does not significantly impact the size of the tree. Also, we can see a clear difference between the memory usage of CSL and the other supervised classifiers after training. (Bottom left) Run times for the training stage. CSL (red) requires roughly an order of magnitude less training run time than SVM (blue). (Bottom right) Average time to recognize a new image after training for the different algorithms. The y axis (logarithmic scale) shows CSL outperforming SVM and KNN by an order of magnitude.

5.1.3. *Memory usage*

The graph in top right of **Figure 5** shows the relationship between the overall number of nodes in the tree to be retained (and hence vectors of dimensionality M) and the branch factor for CSL classifier. CSL with tree descent had to store an average of 1036.25 vectors, while the knn-on-leaves variant had to store 902.21 vectors. SVM required 2286 vectors while the vanilla KNN method (with $k = 1$) requires storage of the entire training corpus of 2322 vectors. Thus, the number of vectors retained in memory by the CSL variants is roughly half the number retained by the SVM and KNN algorithms. Further, the memory needed to store the trained model when we predict using the KNN-on-leaves approach is smaller than when we use tree descent, as we expected and discussed earlier. As can be seen, there is not much variation in CSL performance across different branch factor values. This suggests that after a few initial splits, most of the sub trees have very few categories represented within them and hence the upper bound on the branch factor does not play a significant role in ongoing performance.

5.1.4. *Classifier run times*

The runtime costs of the algorithms paint an even more startling picture. The graph in bottom left of **Figure 5** shows the plots comparing the training times of the CSL and SVM algorithms. The two variants of CSL have the same training procedure and hence require the same time to train. (KNN has no explicit training stage.) As can be seen, the training time of the new algorithm (average of 2.42 s) is roughly an order of magnitude smaller than that of the SVM (average of 18.54 s). It should be clearly noted that comparisons between implementations of algorithms will not necessarily reflect underlying computational costs inherent to the algorithms, for which further analysis and formal treatment will be

required. Nonetheless, in the present experiments, the empirical costs were radically different despite efforts to show the SVM in its best light.

As indicated earlier, the choice of branch factor does not have a large impact on the training time needed. We also found that the working memory requirements of our algorithm were very small compared to that of the SVM. In the extreme, when large representations were used for images, the memory requirements for SVMs rendered the task entirely impracticable. In such circumstances, the CSL method still performed effectively. The working amount of memory we need is proportional to the largest clustering job that needs to be performed. By choosing low values of K^{max}, we empirically find that we can keep this requirement low without loss of classifier performance.

The bottom right plot of **Figure 5** shows how the average time for recognizing a new image varies with branch factor. The times are shown in logarithmic scale. The CSL variants are an order of magnitude faster than KNN and SVM algorithms with the tree descent variant being the fastest. This shows the proposed algorithm in its best light. Once training is complete, recognition can be extremely rapid by doing hierarchical descent, making the CSL method unusually well suited for real-time applications.

5.2. HAXBY fMRI DATASET, 2001

5.2.1. Dataset

Having demonstrated the CSL system on image data, we selected a very different dataset to test: neuroimaging data collected from the brain activity of human subjects who were viewing pictures. As with the Caltech-256 data, we selected a very well-studied set of fMRI data, from a 2001 study by Haxby et al. (2001).

Six healthy human volunteers entered an fMRI neuroimaging apparatus and viewed a set of pictures while their brain activity (blood oxygen-level dependent measures) was recorded. In each run, the subjects passively viewed gray scale images of eight object categories, grouped in 24 s blocks separated by rest periods. Each image was shown for 500 ms and was followed by a 1500-ms interstimulus interval. Each subject carried out twelve of these runs. The stimuli viewed by the subjects consisted of images from the following eight classes: Faces, Cats, Chairs, Scissors, Houses, Bottles, Shoes, and random scrambled pictures. Full-brain fMRI data were recorded with a volume repetition time of 2.5 s, thus, a stimulus block was covered by roughly 9 volumes. For a complete description of the experimental design and fMRI acquisition parameters, see Haxby et al. (2001). (The dataset is publicly available.) Each fMRI recording corresponding to 1volume in a block for a given input image can be thought of as a vector with 163840 dimensions. The recordings for all the subjects have the same vector length. (In the original work, "masks" for individual brain areas were provided, retaining only those voxels that were hypothesized by the experimenters to play a significant role in object recognition. Using these masks reduces the data dimensionality by a large factor. However, the masks are of different lengths for different subjects, thus preventing meaningful aggregation of recordings across subjects. Thus, we have not used the masks and instead trained the classifiers in the original high dimensional space.)

5.2.2. Testing on individual subjects

For each subject who participated in the experiment, we have neuroimaging data collected as that subject viewed images from each of the eight classes. The task was to see whether, from the brain data alone, the algorithms could predict what type of picture the subject was viewing. Top left in **Figure 6** shows the prediction accuracy of the various classifiers we tried. On the whole, all the classifiers exhibit similar performance with SVM performing slightly better on a couple of the subjects.

Top right of **Figure 6** shows the memory requirements for all the algorithms. The CSL variants require significantly less memory to store the model learned during training compared to SVM and KNN. SVM requires a large number of support vectors to fully differentiate the data from different classes leading to large memory consumption, whereas KNN needs to store all the training data in memory. For CSL, if the testing method is tree descent, then the entire hierarchy needs to be kept in memory. For the KNN-on-leaves testing method, only the leaves of the tree are retained, rendering even a smaller memory requirement for the stored model.

Bottom left of **Figure 6** shows the training time for the CSL algorithm being an order of magnitude smaller than that of SVM. KNN does not have any explicit training stage. Finally, bottom right of **Figure 6** compares the recognition time for the different algorithms, again on a log scale. The average recognition time on a new sample for the CSL tree descent variant is a couple of orders of magnitude smaller than both KNN and SVM. For the KNN-on-leaves variant of the CSL method, the recognition time grows larger (while still being significantly smaller than KNN or SVM). Therefore the fastest approach is performing a tree descent (paying a penalty in terms of memory requirements for storing the model).

5.2.3. Aggregating data across subjects

Since the recordings from all the subjects have the same dimensionality, we can merge all the data from the different subjects into 1 large dataset and partition it into the training and testing datasets. This way we can study the performance trends with increasing datasets. The SVM system, unfortunately, was unable to run on pools containing more than two subjects, due to the SVM system's high memory requirements. Nonetheless, the two variants of the CSL algorithm, and the KNN algorithm, ran successfully on collections containing up to five subjects' aggregated data.

The subplot on the left of **Figure 7** shows that the classification prediction accuracy of the different classifiers remain competitive with each other as we increase the pool. The subplot on the right of **Figure 7** shows the trend of memory consumption by the different algorithms as we increase the number of subjects included. Compared to standard KNN, the increase in memory consumption is much slower (sub linear) for the CSL algorithm, with the KNN-on-leaves variant of the CSL algorithm growing very slowly.

Finally, in **Figure 8**, we examine the growth in the average recognition time with increasing pool size. The costs of adding data cause the recognition time to grow for the KNN algorithm more than for either variant of the CSL algorithm (either tree descent or KNN-on-leaves versions).

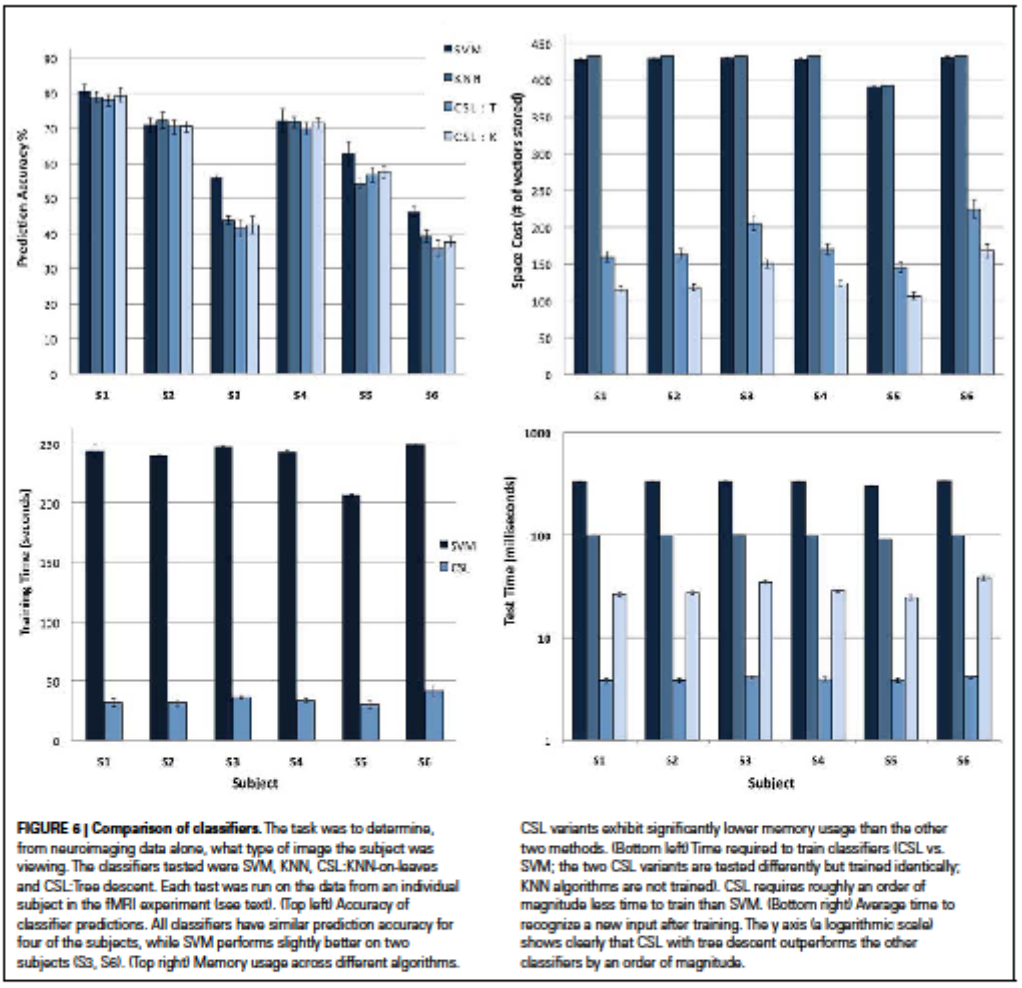


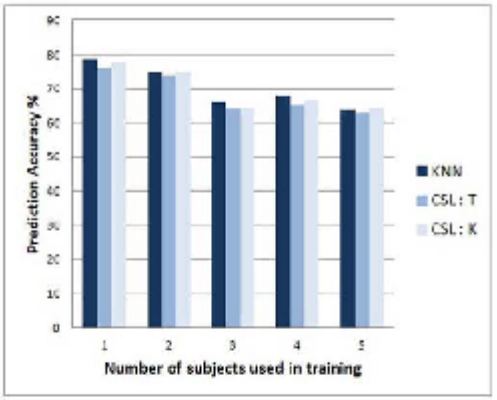
FIGURE 6 | Comparison of classifiers. The task was to determine, from neuroimaging data alone, what type of image the subject was viewing. The classifiers tested were SVM, KNN, CSL:KNN-on-leaves and CSL:Tree descent. Each test was run on the data from an individual subject in the fMRI experiment (see text). (Top left) Accuracy of classifier predictions. All classifiers have similar prediction accuracy for four of the subjects, while SVM performs slightly better on two subjects (S3, S6). (Top right) Memory usage across different algorithms. CSL variants exhibit significantly lower memory usage than the other two methods. (Bottom left) Time required to train classifiers (CSL vs. SVM; the two CSL variants are tested differently but trained identically; KNN algorithms are not trained). CSL requires roughly an order of magnitude less time to train than SVM. (Bottom right) Average time to recognize a new input after training. The y axis (a logarithmic scale) shows clearly that CSL with tree descent outperforms the other classifiers by an order of magnitude.

Between these two CSL algorithm variants, the latter exhibits some modest time growth as data is added, whereas the former (tree descent) version of CSL exhibits no significant increase in recognition time whatsoever as more data is added to the task. It is notable that the reason for this is that the tree depth has not increased with increasing size of the dataset; that is, as more data is added, the learned CSL tree arrives at the ability to successfully classify the data early on, and adding new data does not require the method to add more to the tree. Interestingly, the trees become better balanced as we increase the number of subjects, but their sizes do not increase. The results suggest that the CSL algorithm is better suited to scale to extremely large data sets than either of the competing standard SVM or KNN methods.

6. ANALYSES AND EXTENSIONS

6.1. ALGORITHM COMPLEXITY

When k-means is used for clustering, the time complexity for each partitioning is O(NtK), where N is the number of samples, K is the number of partitions and t is the number of iterations. If we fix t to be a constant (by putting an upper limit on it), then each split takes $O(NK)$. Since we also put a bound on K (K_{max}), we can assume that each split is O(N). Further



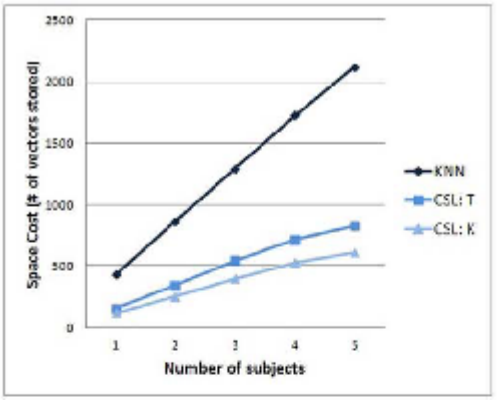


FIGURE 7 | (Left) Accuracy of classifier predictions on split tests. The accuracy of the different algorithms remain competitive as we increase the subject pool. (The memory usage by the LibSVM implementation of SVM was too large for testing on subject pools larger than 2). (Right) Tracks the trend of memory consumption with increasing size of the subject pool for the classifiers. KNN's memory usage grows linearly, whereas the CSL variants grow at a lower rate, illustrating their scalability.

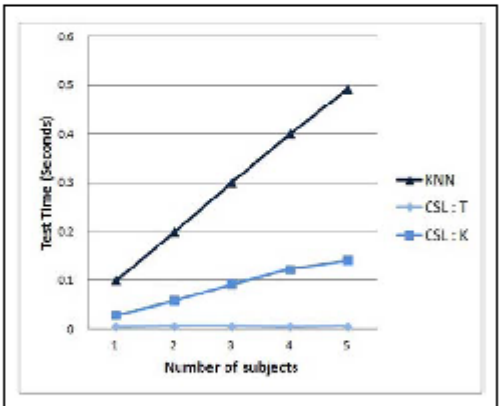


FIGURE 8 | Average recognition time required for a new test sample, as a function of the amount of data trained. As expected, KNN grows linearly. CSL with KNN-on-leaves grows more slowly. Interestingly, CSL with tree descent hardly shows any increase, suggesting its scalability to extremely large datasets.

analysis is needed on the total number of paths and their contribution to runtime. The maximum amount of memory needed is for the first unsupervised partitioning. This is proportional to $O(NK)$. When we have small K, the amount of memory is directly proportional to the number of data elements being used in training.

As mentioned earlier, the algorithm is intrinsically highly parallel. After every unsupervised partitioning, each of the partitions can be further treated in parallel. However, in the experiments reported here, we have as yet made no attempt to parallelize the code, seeking instead to compare the algorithm directly against current standard SVM implementations.

6.2. COMPARISON WITH OTHER HIERARCHICAL LEARNING TECHNIQUES

The structure of the algorithm makes it very similar to CART (and in particular, decision trees; Buntine, 1992) since both families of algorithms partition the non-linear input space into discontinuous regions such that the individual sub regions themselves provide effective class boundaries. However, there are several significant differences.

- Perhaps the most substantial difference is that decision trees use the labels of the data to perform splits, whereas the CSL algorithm partitions based on unsupervised similarity.
- The CSL algorithm splits in a multivariate fashion, taking into account all the dimensions of the data samples, as opposed to decision trees where most often, a single dimension which results in the largest demixing of the data, is used to make splits. The path from the root to a leaf in a decision tree is a conjunction of local decisions on feature values and as a result is prone to over fitting. As discussed before, the CSL tends to exhibit little overfitting, and we can understand why this is the case (see Discussion in the Simplified Algorithm section earlier). The leaves can be treated independently of the rest of the tree and KNN can be used on them to obtain the class predictions.
- Decision trees are by nature a 2 class discriminative approach (multiclass problems can be handled using binary decision trees; Lee and Oh, 2003) whereas the CSL algorithm is a natural multiclass generative algorithm.

Most importantly, the goals of these systems differ. The primary goal of the CSL algorithm is to uncover natural structure within the data. The fact that the label-based impurity of classes is reduced, resulting in the ability to classify labeled data, falls out as a (very valuable) side effect of the procedure. The CSL algorithm thus will carry out a range of additional tasks, beyond supervised classification, that use deeper analysis of the underlying structure of the data, not apparent through supervised labeling alone.

6.3. DISCOVERY OF STRUCTURE

For purposes of this paper we have focused solely on the classification abilities of the algorithm, though the algorithm can perform many other tasks outside the purview of classification. Here we will briefly cover two illustrative additional abilities: (i) uncovering secondary structure of data, and (ii) localization of objects within images.

6.3.1. Haxby dataset

Once a model is trained, for each training sample if we do hierarchical descent and aggregate the posterior probabilities of the nodes along the path, we get a representation for the sample. When we do an agglomerative clustering on that representation, we uncover secondary structure suggesting meta classes occurring in the dataset. **Figure 9** captures the output of such an agglomerative clustering for the recordings of one subject (S1). Here we can see extensive structure relations among the responses to various pictures; perhaps most prominent is a clear separation of the data into animate and inanimate classes. The tree suggests the structure of information that is present in the neuroimaging data; the subjects' brain responses distinguish among the different types of pictures that they viewed. Related results were shown by Hanson et al. (2004); these were arrived at by analysis of the hidden node activity of a back propagation network trained on the same data. In contrast, it is worth noting that the CSL classifier obtains this structure as a natural byproduct of the tree-building process.

6.3.2. Image localization

A task quite outside the realm of supervised classification is that of localizing, i.e., finding an object of interest within an image. This task is useful to illustrate additional capabilities of the algorithm beyond just classification, making use of the internal representations it constructs.

We assume for this example that the clustering component of the algorithm is carried out by a generative method such as PLSA (Sivic et al., 2005); we then can assume that the features specific to the object class will contribute to the way in which an image becomes clustered, and that those features will contribute more than will random background features in the image.

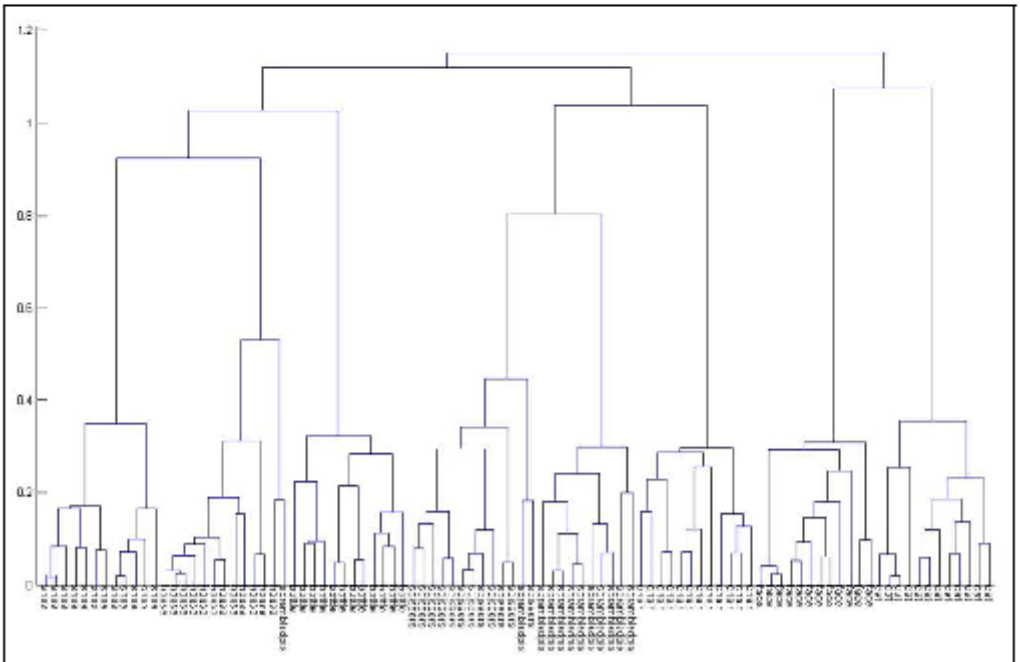


FIGURE 9 | Agglomerative clustering of tree node activity for all the data from Subject 1. The hierarchy shows extensive internal structure in the responses, including a clear separation of animate and inanimate stimuli. See text for further discussion.

Figure 10 shows an example of localization of a face within an image. The initial task was to classify images of faces, cars, motorcycles, and airplanes (from Caltech 4). PLSA was used for clustering and for determining the cluster membership of a previously unseen image x. For every cluster z, we can obtain the posterior probability $p(z|x, w)$, for every feature w in the vocabulary.

FIGURE 10 | An illustration of object localization on an image from Caltech-256. (A) The original image. **(B–E)** Positive, neutral, and negative features (green, blue, red, respectively) shown at levels 1 through 4 along the path in the CSL tree. **(F)** A thresholded map of the aggregate feature scores. Points in green indicate feature scores above threshold and those in red indicate below-threshold scores. Note that although green dots occur in multiple regions, the presence of red dots (negative features) is limited only to regions outside the face region.

Thus, we can test all features in the image to see which ones maximize the posterior, indicating strong influence on the eventual cluster membership. The location of those features can then be used to identify the vicinity of the object.

As the path from root to leaf in the CSL hierarchy is traversed for a particular test image, the posterior at a given node determines the contribution of the feature to the branch selected. Let y be the final object label prediction for image x.

Consider feature f_l from the vocabulary. At any given node at height l along the path leading to prediction of y for x, let d_l^j be the branch predicted by feature f_l, i.e., among all branches at node l, the posterior for that branch is highest for that feature. d_l^j is actually a set of labels that can be reached at various leaves using the branch and finally let the overall branch taken at l be b^j.

At level l, f_l, can be classified as positive if $1(d_l^j == b^j)$, neutral if $1(d_l^j \neq b^j)$ and $1(y \in d_l^j)$, and finally, negative if $1(d_l^j \neq b^j)$ and $1(y \notin d_l^j)$. The overall score for f_l is a weighted sum S_l of all the scores (negative features getting a negative score) along the path. Since we know the locations of the features, we can transfer the scores to actual locations on the images (more than one location may map to the same feature in the vocabulary). When a simple threshold is applied, we get the map seen in the final image. The window most likely to contain the object can then be obtained by optimization of the scores on the map using branch and bound techniques.

7. CONCLUSION

We have introduced a novel, biologically derived algorithm that carries out similarity-based hierarchical clustering combined with simple matching, thus determining when nodes in the tree are to be iteratively deepened. The clustering mechanism is a reduced subset of published hypotheses of thalamocortical function; the match/mismatch operation is a reduced subset of proposed basal ganglia operation; both are described in Granger (2006). The resulting algorithm performs a range of tasks, including identifying natural underlying structure among object in the dataset; these abilities of the algorithm confer a range of application capabilities beyond traditional classifiers. In the present paper we described in detail just one circumscribed behavior of the algorithm: its ability to use its combination of unsupervised clustering and reinforcement to carry out the task of supervised classification. The experiments reported here suggest the algorithm's performance is comparable to that of SVMs on this task, yet requires only a fraction of the resources of SVM or KNN methods.

It is worth briefly noting that the intent of the research described here has not been to design novel algorithms, but rather to educe algorithms that may be at play in brain circuitry. The two brain structures referenced here, neocortex and basal ganglia, when studied in isolation, have given rise to hypothesized operations of hierarchical clustering and of reinforcement learning, respectively (e.g., Sutton and Barto, 1998; Rodriguez et al., 2004). These structures are connected in a loop, such that (striatal) reinforcement learning can be hypothesized to selectively interact with (thalamocortical) hierarchies being constructed. We conjecture that the result is a novel composite algorithm (CSL), which can be thought of as iteratively constructing rich representations of sampled data.

Though the algorithm was conceived and derived from analysis of cortico-striatal circuitry, the next aim was to responsibly analyze its efficacy and costs and compare it appropriately against other competing algorithms in various domains. Thus we intentionally produced very general algorithmic statements of the derived cortico-striatal operations, precisely so that (1) we can retain functional equivalency with the referenced prior literature (Schultz et al., 1997; Suri and Schultz, 2001; Schultz, 2002; Rodriguez et al., 2004; Daw and Doya, 2006); and (2) the derived algorithm can be responsibly compared directly against other algorithms. The algorithm can be applied to a number of tasks; for purposes of the present paper we have focused on supervised classification (though we also briefly demonstrated the utility of the method for different tasks, including identification of structure in data, and localization of objects in an image).

It is not yet known what tasks or algorithms are actually being carried out by brain structures. Brain circuits may represent compromises among multiple functions, and thus may not outperform engineering approaches to particular specialized tasks (such as classification). In the present instance, individual components are hypothesized to perform distinct algorithms, hierarchical clustering and reinforcement learning, and the interactions between those components perform still another composite algorithm, the CSL method presented here. (And, as mentioned, the studied operations are very-reduced subsets of the larger hypothesized operations of these thalamocortical and basal ganglia systems; it is hoped that ongoing study will yield further algorithms arising from richer interactions of these cortical and striatal structures, beyond the reduced simplifications studied in the present paper.) As might be expected of a method that has been selectively developed in biological systems over evolutionary time, these component operations may represent compromises among differential selectional pressures for a range of competing tasks, carried out by combined efforts of multiple distinct engines of the brain. This represents an instance in which models of a biological system lead to derivation of tractable algorithms for real-world tasks. Since the biologically derived method studied here substantially outperforms extant engineering methods in terms of efficacy per time or space cost, we forward the conjecture that brain circuitry may continue to provide a valuable resource from which to mine novel algorithms for challenging computational tasks.

ACKNOWLEDGMENTS

This work was supported in part by grants from the Office of Naval Research and the Defense Advanced Research Projects Agency.

REFERENCES

Alexander, G., and DeLong, M. (1985). Microstimulation of the primate neostriatum. I. Physiological properties of striatal microexcitable zones. *J. Neurophysiol.* 53, 1401–1416.

Arthur, D., and Vassilvitskii, S. (2007). "k-Means++: the advantages of careful seeding," in *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans.

Baron, A., and Cover, T. (1991). Minimum complexity density estimation. *IEEE Trans. Inf. Theory* 4, 1034–1054.

Bishop, C. (1996). *Neural Networks for Pattern Recognition*. New York: Oxford University Press.

Blei, D., Ng, A., and Jordan, M. (2003). Latent Dirichlet allocation. *J. Mach. Learn. Res.* 3, 993–1022.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

Brown, J., Bullock, D., and Grossberg, S. (1999). How the basal ganglia use parallel excitatory and inhibitory learning pathways to selectively respond to unexpected reward. *J. Neurosci.* 19, 10502–10511.

Buntine, W. (1992). Learning classification trees. *Stat. Comput.* 2, 63–73.

Chang, C., and Lin, C. (2001). *Libsvm: A Library for Support Vector Machines*. Available at: http://www.csie.ntu.edu.tw/cjlin/libsvm

Darken, C., and Moody, J. (1990). "Fast, adaptive k-means clustering: some empirical results," in *Proceedings of the IEEE IJCNN Conference* (San Diego: IEEE Press).

Daw, N. (2003). *Reinforcement Learning Models of the Dopamine System and Their Behavioral Implications*. Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA.

Daw, N., and Doya, K. (2006). The computational neurobiology of learning and reward. *Curr. Opin. Neurobiol.* 16, 199–204.

Felch, A., and Granger, R. (2008). The hypergeometric connectivity hypothesis: divergent performance of brain circuits with different synaptic connectivity distributions. *Brain Res.* 1202, 3–13.

George, D., and Hawkins, J. (2009). Towards a mathematical theory of cortical microcircuits. *PLoS Comput. Biol.* 5, e1000532. doi:10.1371/journal.pcbi.1000532

Granger, R. (2006). Engines of the brain: the computational instruction set of human cognition. *AI Mag.* 27, 15–32.

Granger, R. (2011). *How Brains are Built: Principles of Computational Neuroscience*. Cerebrum: The Dana Foundation, Available at: http://dana.org/news/cerebrum/detail.aspx?id=30356.

Griffin, G., Holub, A., and Perona, P. (2007). *Caltech-256 Object Category Dataset*. California Institute of Technology.

Gurney, K., Prescott, T., and Redgrave, P. (2001). A computational model of action selection in the basal ganglia: I. a new functional anatomy. *Biol. Cybern.* 84, 401–410.

Hanson, S., Matsuka, T., and Haxby, J. (2004). Combinatorial codes in ventral temporal lobe for object recognition: Haxby (2001). Revisited: is there a face area? *Neuroimage* 23, 156–166.

Haxby, J., Gobbini, M., Furey, M., Ishai, A., Schouten, J., and Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science* 293, 2425–2430.

Hoffman, T. (1999). "Probabilistic latent semantic indexing," in *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, 50–57.

Houk, J., Bastianen, C., Fansler, D., Fishbach, A., Fraser, D., Reber, P., Roy, S., and Simo, L. (2007). Action selection and refinement in subcortical loops through basal ganglia and cerebellum. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 362, 1573–1583.

Hsu, C., and Lin, C. (2002). A comparison of methods for multi-class support vector machines. *IEEE Trans. Neural Netw.* 13, 415–425.

Kaski, S. (1997). Data exploration using self-organizing maps. *Acta Polytechnica Scand. Math. Comput. Manag. Eng. Ser.* 82.

Kemp, J., and Powell, T. (1971). The connexions of the striatum and globus pallidus: synthesis and speculation. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 262, 441–445.

Kosko, B. (1991). Stochastic competitive learning. *IEEE Trans. Neural Netw.* 2, 522–529.

Leblois, A., Borand, T., Meissner, W., Bergman, H., and Hansel, D. (2006). Competition between feedback loops underlies normal and pathological dynamics in the basal ganglia. *J. Neurosci.* 26, 3567–3583.

Lee, J., and Oh, I. (2003). "Binary classification trees for multi-class classification problems" in *ICDAR '03 Proceedings of the Seventh International Conference on Document Analysis and Recognition*, Edinburgh.

Lee, T., and Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *Opt. Soc. Am.* 20, 1434–1448.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Trans. Inf. Theory* 28, 129–136.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* 60, 91–110.

Marr, D. (1980). *Vision*. MIT press.

McGeorge, A., and Faull, R. (1988). The organization of the projection from the cerebral cortex to the striatum in the rat. *Neuroscience* 29, 503–537.

Ng, A., and Jordan, M. (2002). On discriminative vs. generative classifiers. a comparison of logistic regression and naive Bayes. *Neural Inf. Process. Syst.* 2, 841–848.

O'Doherty, J., Dayan, P., Friston, K., Critchley, H., and Dolan, R. (2003). Temporal difference models and reward-related learning in the human brain. *Neuron* 38, 329–337.

Riesenhuber, M., and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nat. Neurosci.* 2, 1019–1025.

Rodriguez, A., Whitson, J., and Granger, R. (2004). Derivation and analysis of basic computational operations of thalamocortical circuits. *J. Cogn. Neurosci.* 16, 856–877.

Schultz, W. (2002). Getting formal with dopamine and reward. *Neuron* 36, 241–263.

Schultz, W., Dayan, P., and Montague, R. (1997). A neural substrate of prediction and reward. *Science* 175, 1593–1599.

Sivic, J., Russell, B., Efros, A., Zisserman, A., and Freeman, W. (2005). Discovering objects and their location in images. *IEEE Int. Conf. Comput. Vis.* 1, 370–377.

Sivic, J., and Zisserman, A. (2003). "Video Google: a text retrieval approach to object matching in videos," in *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, Nice.

Stephan, H. (1972). "Evolution of primate brains: a comparative anatomical approach," in *Functional and Evolutionary Biology of Primates*, ed. R. Tuttle (Chicago: Aldine-Atherton), 155–174.

Stephan, H., Bauchot, R., and Andy, O. (1970). Data on size of the brain and of various brain parts in insectivores and primates. *Advances in Primatology*, 289–297.

Stephan, H., Frahm, H., and Baron, G. (1981). New and revised data on volumes of brain structures in insectivores and primates. *Folia Primatol.* 35, 1–29.

Suri, R., and Schultz, W. (2001). Temporal difference model reproduces anticipatory neural activity. *Neural Comput.* 13, 841–862.

Sutton, R., and Barto, A. (1990). "Time derivative models of Pavlovian reinforcement," in *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, eds. M. Gabriel, and J. Moore (MIT Press), 497–537.

Sutton, R., and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT press.

Teh, Y., Jordan, M., Beal, M., and Blei, D. (2004). "Sharing clusters among related groups: hierarchical dirichlet processes," in *Proceedings of Neural Information Processing Systems*, Vancouver.

Ullman, S. (2006). Object recognition and segmentation by a fragment-based hierarchy. *Trends Cogn. Sci. (Regul. Ed.)* 11, 58–64.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 27 May 2011; accepted: 28 October 2011; published online: 10 January 2012.

Citation: Chandrashekar A and Granger R (2012) Derivation of a novel efficient supervised learning algorithm from cortical-subcortical loops. Front. Comput. Neurosci. 5:50. doi: 10.3389/fncom.2011.00050

This article was submitted to Frontiers in Iterative computations of cortico-striatal loops, a specialty of Frontiers in Computational Neuroscience.

Appendix B

Dartmouth Brain Engineering Laboratory
BELAB Technical Report 2012-3

Learning what is where from unlabeled images:

Joint localization and clustering of foreground objects

Ashok Chandrashekar · Lorenzo Torresani · Richard Granger

Abstract "What does it mean, to see? The plain man's answer would be, to know what is where by looking." This famous quote by David Marr [21] sums up the holy grail of vision: discovering what is present in the world, and where it is, from unlabeled images. In this paper we tackle this challenging problem by proposing a generative model of object formation and describe an efficient algorithm to automatically learn the parameters of the model from a collection of unlabeled images. Our algorithm discovers the objects and their spatial extents by clustering together images containing similar foregrounds. Our approach simultaneously solves for the image clusters, the foreground appearance models and the spatial regions containing the objects by optimizing a single likelihood function defined over the entire image collection. We describe two novel methods for efficient foreground localization: the first method does not require any bottom-up image segmentation and discovers the foreground region as a contiguous rectangular bounding box. The second method expresses the foreground as a collection of super pixels generated through a bottom-up segmentation of the image. However, unlike previous methods, objects are not assumed to be encapsulated by a single segment. Evaluation on standard benchmarks and comparison with prior methods demonstrate that our approach achieves state-of-the-art results on the problem of unsupervised foreground localization and clustering.

Keywords Unsupervised · Clustering · Foreground · Localization

Ashok Chandrashekar
Dartmouth College
E-mail: ashok@cs.dartmouth.edu

Lorenzo Torresani
Dartmouth College
E-mail: lorenzo@cs.dartmouth.edu

Richard Granger
Dartmouth College
E-mail: richard.granger@dartmouth.edu

This work was supported in part by grants from the Defense Advanced Projects Agency and from the Office of Naval Research.

1 Introduction

Object categorization requires recognizing the classes of objects appearing in an input photo. Rather than performing classification of the entire image as a whole, object class recognition systems often operate by decomposing the photo into different regions corresponding to the objects present in the scene. Treating object localization and recognition jointly allows such methods to be more robust to clutter, variations in backgrounds, as well as presence of multiple objects.

We can distinguish several methodologies for object recognition and localization on the basis of the amount of human supervision needed during training. When the training images are manually segmented into semantic regions, object localization can be formulated as the task of densely matching regions of the input photo to the manually annotated segments of similar images in the database [19]. In order to achieve good results, these methods require very large collections of annotated images so as to maximize the chance of a close image match in the database. However, due to the cost of collecting pixel-labels, such datasets are extremely time-consuming to generate and difficult to label accurately.

A second methodology involves the use of datasets where only the object of interest is manually segmented in the training images. Typically, recognition and localization is then achieved using a combination of bottom-up segmentation and top down classification ([5],[18],[28],[30]). But these methods are computationally expensive to run and, again, the requirement for detailed segmentation in the training set is far too onerous.

An efficient alternative is object detection([7],[8]), which involves sliding a subwindow classifier exhaustively over all rectangular regions of the test image in order to robustly localize the box that is most likely to contain the object. This brute-force evaluation can be made very efficient by using a branch and bound strategy [16] which allows to rapidly remove from consideration a large portion of regions. These algorithms normally require the object to be delineated using a bounding box in the training dataset, which is easier to generate compared to full segmentation. However, even this form of labeling is expensive to acquire and effectively restricts the size of the training set. Furthermore, the sizes and locations of the bounding boxes are typically chosen arbitrarily by the labeler and are consequently unlikely to be optimal for recognition.

When images have labels indicating the objects present in them but no locality information for the objects, semi supervised methods can be applied to learn automatically the correspondences between image regions and the labels of the image. Most methods in this genre use bottom-up segmentation as a preprocessing to produce candidate segments, and then perform top down learning on the segments ([10],[2],[6]). However the main weakness in such methods is relying on the ill defined task of bottom-up segmentation (based on low-level visual cues such as edges and texture) to segment images such that objects or semantically-coherent regions are represented by a single segment. Thus, such approaches typically yield poor classification accuracy. Recently, Nguyen et al.[22] and Deselaers et al.[9] have proposed weakly-supervised object localization methods avoiding the need of

bottom-up segmentation: the idea of these methods is to simultaneously localize discriminative subwindows in the training images and to learn a classifier to recognize such regions. However, even such methods require supervision in terms of class labels.

In this paper we contrast the traditional methodologies for object localization and recognition outlined above, by presenting a fully-unsupervised method which completely eliminates the need for time-consuming and suboptimal human labeling. The intuition behind our approach is that objects can be viewed as recurring foreground patterns appearing as coherent image regions. Thus, we can formulate object discovery as the task of partitioning an unlabeled collection of images into K subsets (clusters), such that all images within each subset share a similar foreground. In order to obtain a method scalable to large collections and many classes, we adopt a foreground mask-based representation of objects, which enables fast localization given the object model. Specifically, we represent the object in an image as a histogram of quantized local features occurring in the enclosing foreground mask. We view each object instance as a random variable drawn from an unknown distribution common to all instances of that object class. This common distribution assumption constrains all foreground histograms of an object class to represent subtle variations around a prototypical average histogram. Based on this assumption, our approach poses object discovery as a maximum likelihood estimation problem, to be optimized over the entire collection of unlabeled images. We present a method that maximizes this objective by simultaneously solving for the histogram model parameters of the object classes, detecting the object instances of each class in the unlabeled images, and performing a soft semantic clustering of images in the data set. In the next section we review prior methods for unsupervised object discovery and discuss their relation to our approach.

2 Related work

Class-generic methods for object discovery, such as [1] and [14], attempt to discover image regions which are strong candidates for containing objects in them. These methods operate on individual images in a purely bottom up fashion. However, the bottom up notion of 'object'ness is ill-defined and hence methods which can discover objects by using a collection of images by determining statistically reoccurring image fragments are more likely to succeed at the task.

Lee and Grauman [17] have proposed an approach to automatically localize foreground features from a collection of unlabeled images. By learning the 'significance' weights of semi-local features iteratively through image grouping, their method determines for each image which features are most relevant, given the image content in the remainder of the collection. While this work successfully demonstrates that a mutual reinforcement of object-level and feature-level similarity improves unsupervised image clustering, there is no clear way of translating feature weights into foreground localization and object extents. Furthermore, it performs clustering from pairwise image matches and therefore the computational cost at each iteration is cubic in number of images. Finally, the algorithm alternates between image clustering and updating the foreground weights without a

unifying formal objective and thus its convergence properties are unclear.

Various semantic topic models ([11],[27], [12],[15], [9]) have been have proposed for similar tasks where the location of the object is treated as a latent variable to be estimated. However, most of these methods are not fully unsupervised and often resort to an expensive sliding window mechanism for object discovery with unknown costs for detection.

Our work is inspired by the approach of Russell et al.[24], who extend their earlier work [26] and propose a fully-unsupervised algorithm to discover objects and associated segments from a large collection of images. Multiple segmentations are performed for each image by varying the parameters of a segmentation method. The key assumption is that each object instance is correctly segmented (as a single contiguous segment) at least once through multiple segmentation and therefore the correct segments corresponding to object classes occur more often than random background. This suggests that the features of correct segments form object-specific coherent clusters discoverable using latent topic models from text analysis. Although the algorithm is shown to be able to discover many different objects, it still suffers from its dependence on bottom-up segmentation to come up with a single segment encapsulating the object, which is ill-posed particularly in the case of unsupervised datasets since often it is necessary to know the category of the object in order to reliably segment it from the scene. Their goal is different from ours in that their method does not prescribe a way to cluster the images or determine which regions in the images correspond to image foregrounds. Nevertheless, in the experiments we consider adaptations of their method to our task for a quantitative comparison.

In contrast, we propose a generative model of foreground formation that enables simultaneous image clustering and foreground localization via maximum likelihood estimation. Unlike [24], our approach treats each image as a composition of foreground and background where the foreground is explained by a single model shared with other images and the background is image-specific and hence not modeled. We treat the foreground mask as a parameter to be estimated as part of the likelihood optimization. We demonstrate that this leads to better localization and image clustering. Apart from the proposed unified framework of maximum likelihood estimation for the task, the main contributions of this paper are the development of two novel methods for efficient localization of object foregrounds in images. In the first method, the foreground is encapsulated by a rectangular bounding box thus obviating the need for bottom-up segmentation. The second method does rely on bottom-up segmentation. However, the segments generated are assumed to be nothing more than 'super-pixels'. In particular, we do not assume that the foreground is captured by a single segment. Hence, we overcome most of the drawbacks of previous methods which tend to generalize poorly due to their reliance on the assumption that bottom-up segmentation will likely produce object instance segments consistently across images belonging to the same class.

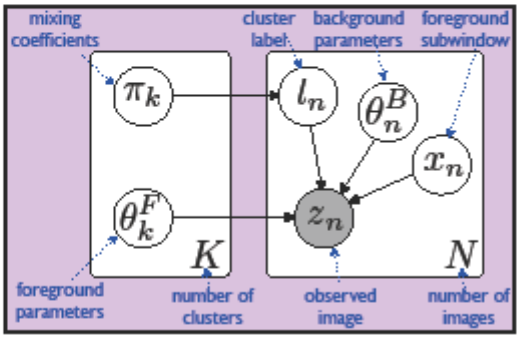


Fig. 1: Our generative model of image formation: image z_n is obtained by first drawing its object class (l_n); then the appearance of the object inside the foreground location (x_n) is generated from a distribution ($\theta^F_{l_n}$) common to all objects instances of that class. The background model (θ^B_n) is assumed to change with every image.

3 Generative model for unsupervised object discovery

We now describe our proposed generative model for unsupervised object discovery. We assume we are given as input a collection of N unlabeled images $z_1, \ldots, z_N$, with each image containing one of K objects. Our objective is twofold: to separate the images into K disjoint subsets (clusters) corresponding to the K object classes and to localize the object within each image. We denote with x_n the unknown foreground mask enclosing the foreground object of image z_n. We represent the foreground region x_n of image z_n by computing the un-normalized histogram $h(z_n, x_n) \in \mathbb{N}^W$ of the visual words (i.e., quantized local visual features) occurring inside x_n: here W represents the number of unique words in the visual codebook, which, as usual, is learned during an offline stage from training images. We assume that the foreground histograms of images belonging to the k-th object class are generated from a common model defined by parameters θ^F_k. Specifically, let $l_n \in \{1, \ldots, K\}$ denote the unknown cluster label of image z_n, which we assume to be drawn from a Multinomial distribution with parameters $\pi = \{\pi_1, \ldots, \pi_K\}$. Then, we model the foreground histogram $h(z_n, x_n)$ as a random variable drawn from a Gaussian distribution with parameters $\theta^F_{l_n} = \{\mu_{l_n}, \Sigma_{l_n}\}$, i.e., $h(z_n, x_n) \sim \mathcal{N}(\mu_{l_n}, \Sigma_{l_n})$. In order to reduce the number of parameters to be estimated, we assume the covariance Σ_k of each cluster k to be diagonal: $\Sigma_k = \mathrm{diag}(\lambda_{k1}, \ldots, \lambda_{kW})^{-1}$. Finally, each image is assumed to have its own independent background model defined by parameters θ^B_n. For our objective of object discovery, the background parameters can be left unresolved. The complete generative model is summarize graphically in Figure 1. We propose to maximize the likelihood of this model by marginalizing over the cluster

labels, which we treat as hidden variables. In other words, our objective is to find parameters $\theta = \{\theta^F, \pi\}$ and foreground regions $x = \{x_1, \ldots, x_n\}$ maximizing

$$p(z|x,\theta)p(x) = \prod_{n=1}^{N} p(z_n|x_n,\theta)p(x_n) = \prod_{n=1}^{N}\sum_{k=1}^{K} p(z_n, l_n = k|x_n,\theta)p(x_n) \quad (1)$$

where $p(x_n)$ is a prior penalizing unlikely configurations of the foreground mask.

4 Optimization

We can maximize the proposed penalized likelihood via an Expectation Maximization (EM) algorithm alternating between estimating the distribution over the cluster labels l_n and solving for the foreground models and locations. Next, we show how to perform each of these steps and demonstrate that our modeling choices lead to efficient localization of the object regions given the foreground parameters θ. The penalized complete log-likelihood of our model is given by:

$$\begin{aligned}\mathcal{L} &= \log \prod_{n=1}^{N} p(z_n, l_n|x_n,\theta)p(x_n) \\ &= \log \prod_{n=1}^{N} p(z_n|l_n, x_n,\theta)p(l_n|\theta)p(x_n) \\ &= \sum_{n=1}^{N} \log p(z_n|x_n, l_n,\theta) + \log p(l_n|\theta) + \log p(x_n) \end{aligned} \quad (2)$$

The E-step of the algorithm involves calculating the latent posterior distribution $\gamma_{nk} \equiv p(l_n = k|z_n, x_n, \theta)$ given the current estimates for θ and x. It can be seen that this reduces to an evaluation of the following equation:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(h(z_n, x_n); \mu_k, \Sigma_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(h(z_n, x_n); \mu_{k'}, \Sigma_{k'})} \quad (3)$$

The M-step requires maximizing the expected log-likelihood $< \mathcal{L}(\theta) >_\gamma$ with respect to θ and x. We begin by writing the expected log likelihood:

$$\begin{aligned}< \mathcal{L} >_\gamma = &\sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk} \log \mathcal{N}(h(z_n, x_n); \mu_k, \Sigma_k) \\ &+ \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk}\pi_k + \sum_{n=1}^{N} \log p(x_n) + \text{const}\end{aligned} \quad (4)$$

The update steps for parameters θ can be obtained by setting the respective derivatives to zero. This leads to the following rules:

$$\pi_k \leftarrow \frac{1}{N}\sum_{n=1}^{N}\gamma_{nk} \tag{5}$$

$$\mu_k \leftarrow \frac{1}{\sum_{n=1}^{N}\gamma_{nk}}\sum_{n=1}^{N}\gamma_{nk}h(z_n,x_n) \tag{6}$$

$$\lambda_{kw}^{-1} \leftarrow = \frac{1}{\sum_{n=1}^{N}\gamma_{nk}}\sum_{n=1}^{N}\gamma_{nk}([h(z_n,x_n)]_w - [\mu_k]_w)^2 \tag{7}$$

where $[a]_w$ denotes the w-th entry of a vector a.

In the M-step we also need to update the estimate of the foreground mask x_n by solving the following optimization:

$$\begin{aligned}\arg\max_{x_n} < \mathcal{L} >_\gamma &= \arg\max_{x_n}\{\log p(x_n) + \sum_{k=1}^{K}\gamma_{nk}\log\mathcal{N}(h(z_n,x_n);\mu_k,\Sigma_k)\} \\ &= \arg\max_{x_n}\{\log p(x_n) - \sum_{k=1}^{K}\gamma_{nk}\sum_{w=1}^{W}\lambda_{kw}([h(z_n,x_n)]_w - [\mu_k]_w)^2\}\end{aligned} \tag{8}$$

We now show that this objective can be rewritten in a form that leads to efficient optimization. Let $\lambda_k = [\lambda_{k1},\ldots,\lambda_{kW}]^T \in \mathbb{R}^W$, $c = [\gamma_{n1}\lambda_1^T,\ldots,\ldots\gamma_{nK}\lambda_K^T]^T \in \mathbb{R}^{WK}$, $\hat{\mu} = [\mu_1^T,\ldots,\mu_K^T]^T \in \mathbb{R}^{WK}$, and finally let us denote with $\hat{h}(x_n)$ the vector containing K copies of $h_n(z_n,x_n)$, i.e., $\hat{h}(x_n) = [h(z_n,x_n)^T,\ldots,h(z_n,x_n)^T]^T \in \mathbb{R}^{WK}$. Then, we can rewrite the objective of eq. 8 equivalently as follows:

$$\arg\max_{x_n} < \mathcal{L} >_\gamma = \arg\max_{x_n\in\mathcal{X}}\{\log p(x_n) - \sum_{j=1}^{KW} c_j([\hat{h}(x_n)]_j - [\hat{\mu}]_j)^2\} \tag{9}$$

We next introduce methods to optimize this objective efficiently.

4.1 Image foregrounds as rectangular bounding boxes

A popular way for circumscribing an object in an image is by using rectangular bounding boxes. Traditionally for the object detection task, the bounding boxes are determined using an expensive sliding window method ([7], [8]). However, recently Lampert et al. [16] have introduced a branch and bound optimization procedure to localize bounding boxes efficiently. In our first proposed approach for determining foreground locality, we treat the foreground of each image as a contiguous rectangular region which is represented by the variable $x_n \in \mathcal{X}$. Here $\mathcal{X}$ indicates the space of all rectangular subwindows. The foreground content $h(z_n,x_n)$ is just a histogram of all features that occur within the rectangle.

Consider eq. 9: note that the second term in this objective is a weighted Euclidean distance between $\hat{\mu}$ and the histogram $\hat{h}(x_n)$ computed from the visual words in subwindow x_n. For such term, we can define a quality lower bound function over sets of subwindows as described by Lampert et al.[16]. For simplicity, let us denote $[\hat{h}(x_n)]_j$ and $[\hat{\mu}]_j$ as $h(x)_j$ and μ_j respectively. Let x^{min} and x^{max}

be the smallest and largest rectangles in candidate set $\mathcal{X}$. We observe that the value of each histogram bin over a set of rectangles $\mathcal{X}$ can be bounded from below and from above by the number of features with corresponding cluster index that fall into x^{min} and x^{max} respectively. We denote these bounds by $\underline{h}(x)_j$ and $\overline{h}(x)_j$ respectively. Each summand can now be bounded from below by

$$c_j(h(x)_j - \mu_j)^2 \geq \begin{cases} c_j(\underline{h}(x)_j - \mu_j)^2 & \text{if } \mu_j < \underline{h}(x)_j \\ 0 & \text{if } \underline{h}(x)_j \leq \mu_j \leq \overline{h}(x)_j \\ c_j(\overline{h}(x)_j - \mu_j)^2 & \text{if } \mu_j > \overline{h}(x)_j \end{cases} \quad (10)$$

In our implementation, we model the first term, $p(x_n)$ as a simple 2D Gaussian over the relative width and height of the foreground subwindow, measured as fractions of the image width and height. Therefore, the bound over sets of subwindows can be trivially defined for $\log p(x_n)$. This implies that our complete objective can now be globally optimized over $x_n \in \mathcal{X}$ using the branch and bound method for efficient subwindow search of [16].

4.2 Image foregrounds as a set of super pixels

Modeling foregrounds as rectangular regions forces foregrounds to be rigid and contiguous. In some cases, this results in the inclusion of random background clutter as part of the window which influences the foreground object model. This is undesirable and is particularly troublesome for highly contoured objects and object classes with large pose variance. To address this concern, we propose a second method of representing foregrounds. Here, each image z_n, undergoes bottom-up segmentation once at the start of the clustering procedure and is split into a number of appearance-based segments $\{s_n^1, s_n^2 ... s_n^M\}$. The number of segments, M, is large enough for the image to be deemed as over segmented. These segments are called super pixels. Thus, the goal of finding the foreground becomes equivalent to finding which superpixels may be part of the foreground. An important property of considering an image as a collection of super pixels is that unlike [24],[10] and several other approaches, we do not require that the entire foreground object region be captured by a single bottom-up segment. Instead we treat the foreground to be composed of a group of super pixels.

Formally, the foreground mask x_n from figure 1 is described by a sequence of variables $\{x_n^1, x_n^2 ... x_n^M\}$. We treat each x_n^i as a variable such that $x_n^i \in [0, 1]$, with the interpretation that higher values imply that the super pixel s_n^i is to be part of the foreground region and a value close to 0 implies that s_n^i is assigned as part of the background. Therefore the foreground image content is defined as $h(x_n) = \sum_i x_n^i h(x_n^i)$, where $h(x_n^i)$ is the histogram of features occurring in the super pixel s_n^i. Using this, we recast eq. 9 as

$$\arg\max_{x_n} < \mathcal{L} >_\gamma = \arg\min_{x_n} \{ \sum_{f,g \in S_N} (x_n^f - x_n^g)^2 + \sum_{j=1}^{KW} c_j([\sum_{i=1}^{M} x_n^i \hat{h}(x_n^i)]_j - [\hat{\mu}]_j)^2 \} \quad (11)$$

$$\text{subject to:} \quad x_n^i \in [0,1], \quad \sum_i x_n^i \frac{P_n^i}{P_n} \in [a,b] \tag{12}$$

where S_N is the set of all pairs of neighboring segments[1] in image z_n, P_n^i is the number of pixels in segment s_n^i and $P_n = \sum_i P_n^i$, and a, b are scalar values constraining the size of object foregrounds (in our experiments we set $a = 0.25, b = 0.75$). The first term in eq. 11 represents our choice of prior $p(x_n)$. This term penalizes configurations where neighboring segments have widely differing values and thus forces foreground segments to be localized together. It can be seen that the eq. 11 is a simple convex optimization objective when x_n^i is allowed to be a real value and hence can be minimized very efficiently using quadratic programming.

5 Implementation details

5.1 Image representation

Our representation is based on histograms of quantized SIFT features [20]. We experimented with both SIFT descriptors calculated densely over the entire image and also those produced using an interest point detector. Similarly to what reported by the authors in [17], we obtained better results using dense descriptors calculated at every pixel in the image. Thus, here we present experiments based only on dense features. As per common practice, we quantize the SIFT descriptors using a vocabulary of visual words generated by running k-means on a set of SIFT descriptors obtained from the collection of input images. We then learn a codebook of LDA topics [4] learned over the quantized SIFT features via Gibbs Sampling [13]. Therefore, each image is viewed as a document of visual words generated from a mixture of topics and the final histogram is produced by assigning each quantized SIFT descriptor to its most likely topic. In our experiments we determined that histograms over a small LDA codebook provided the most consistent results.

5.2 Initialization

The method of initialization for unsupervised clustering often has a large impact on the quality of the final results. The parameters to initialize in our model are: mixture coefficients (π_k), histogram means (μ_k) and variances (Σ_k), and foreground masks for all images (x_n).
We have evaluated bottom up, class-generic object detectors such as [1] for suitability for getting an initial estimate of image foregrounds. However, in practice such methods are unreliable. Therefore, we have developed a novel approach to initialize object foreground locations. We essentially perform a pairwise foreground matching for all images in the dataset. For each pair of images, we find the two foreground masks that minimize the L1-norm distance between histograms computed from these masks. This can be viewed as a form of co-segmentation [23], aimed at finding the most similar subwindows in the two images. Specifically, for

[1] Note that, since the segments do not change after the initial image segmentation, neither do neighborhood relationship between segments.

$$\text{subject to:} \quad x_n^i \in [0,1], \quad \sum_i x_n^i \frac{P_n^i}{P_n} \in [a,b] \tag{12}$$

where S_N is the set of all pairs of neighboring segments[1] in image z_n, P_n^i is the number of pixels in segment s_n^i and $P_n = \sum_i P_n^i$, and a, b are scalar values constraining the size of object foregrounds (in our experiments we set $a = 0.25, b = 0.75$). The first term in eq. 11 represents our choice of prior $p(x_n)$. This term penalizes configurations where neighboring segments have widely differing values and thus forces foreground segments to be localized together. It can be seen that the eq. 11 is a simple convex optimization objective when x_n^i is allowed to be a real value and hence can be minimized very efficiently using quadratic programming.

5 Implementation details

5.1 Image representation

Our representation is based on histograms of quantized SIFT features [20]. We experimented with both SIFT descriptors calculated densely over the entire image and also those produced using an interest point detector. Similarly to what reported by the authors in [17], we obtained better results using dense descriptors calculated at every pixel in the image. Thus, here we present experiments based only on dense features. As per common practice, we quantize the SIFT descriptors using a vocabulary of visual words generated by running k-means on a set of SIFT descriptors obtained from the collection of input images. We then learn a codebook of LDA topics [4] learned over the quantized SIFT features via Gibbs Sampling [13]. Therefore, each image is viewed as a document of visual words generated from a mixture of topics and the final histogram is produced by assigning each quantized SIFT descriptor to its most likely topic. In our experiments we determined that histograms over a small LDA codebook provided the most consistent results.

5.2 Initialization

The method of initialization for unsupervised clustering often has a large impact on the quality of the final results. The parameters to initialize in our model are: mixture coefficients (π_k), histogram means (μ_k) and variances (Σ_k), and foreground masks for all images (x_n).
We have evaluated bottom up, class-generic object detectors such as [1] for suitability for getting an initial estimate of image foregrounds. However, in practice such methods are unreliable. Therefore, we have developed a novel approach to initialize object foreground locations. We essentially perform a pairwise foreground matching for all images in the dataset. For each pair of images, we find the two foreground masks that minimize the L1-norm distance between histograms computed from these masks. This can be viewed as a form of co-segmentation [23], aimed at finding the most similar subwindows in the two images. Specifically, for

[1] Note that, since the segments do not change after the initial image segmentation, neither do neighborhood relationship between segments.

each pair of images (z_i, z_j), we find the pair of subwindows $(x_{ij}, x_{ji}) \in \mathcal{X} \times \mathcal{X}$ that minimizes the following objective:

$$||h(z_i, x_{ij}) - h(z_j, x_{ji})||_1 - C||h(z_i, x_{ij}) + h(z_j, x_{ji})||_1 \quad (13)$$

where $||.||_1$ denotes the L1-norm and C is a hyperparameter trading off the objectives of finding similar histograms and choosing large subwindows (in our implementation C is set to 0.2). It is easy to see that this objective can be minimized using a simple variant of the branch and bound method described in [16]. At the end of this pairwise matching process, for each image z_i, we get $N-1$ candidate foreground masks $\{x_{ij}\}_{j \neq i}$. From this set, we pick the 3rd largest window by area. The intuition behind this choice is that close matches will result in larger windows and that the largest windows probably contain background regions due to matching to near-duplicates. The same initial windows were used for both the foreground localization methods described in this paper. While it is true that the cost of initialization is quadratic in the number of images, we stress that its a one time cost unlike most competing methods (such as [17]), where each iteration has a quadratic cost. Once initialized, the iterations of our EM algorithm have linear cost.

For initializing the mixture parameters, we tried a variation of careful seeding [3] which was robust against outliers.

6 Experimental results

There are very few published quantitative evaluations on the task of unsupervised clustering and foreground localization. In this paper, we benchmark the performance of our proposed method principally against the results published in [17] (FF), which reports on the same task. We do not compare directly to the methods described in [29] as these algorithms do not consider the problem of object localization and instead perform image clustering merely based on global features calculated from the entire image. Instead we include as baselines a mixture of gaussians model applied to whole images (GMM-whole) as well as ground truth bounding boxes (GMM-GT), to show the benefits provided by our foreground localization methods in the clustering results. We have also applied the gaussian mixture model on bounding boxes derived using the bottom up method described in [1] (GMM-Obj). Finally, we have interpreted the method described in [24] (Multi-Seg) and applied it to our task.

In [17], the authors have evaluated their method on the MSRC-v1 dataset and two subsets (a 4-class and a 10-class collection) of the Caltech 101 dataset. Please refer to that paper for details on the datasets. Here we report our findings using exactly the same experimental setup and sets of images. For all datasets, we pick the number of foreground clusters, K, to be equal to the number of classes. We performed all experiments using a codebook of 50 LDA topics computed from 500 SIFT words. For the segment selection method of foreground localization, we generate 20 bottom-up segments for every image using an implementation of normalized cuts [25].

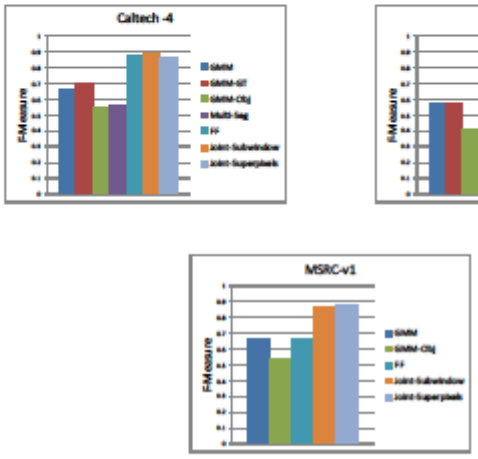


Fig. 2: The quality of image clustering in terms of the F-Measure metric for the three datasets. The compared methods are GMM applied to full images (GMM-Whole), ground truth subwindows (GMM-GT) and object boxes derived using [1] (GMM-Obj). The plots also include results for Multi-Seg[24], FF[17], and our proposed algorithm of joint clustering and localization (subwindow discovery as well as segment selection)

6.1 Quality of image clustering

We begin by evaluating the quality of clustering as F-measure metric with respect to the ground truth class labels: $F = \sum_i \frac{N_i}{N} max_j F'(i,j)$, where N_i is the number of images belonging to class i, $F'(i,j) = \frac{2R(i,j)P(i,j)}{R(i,j)+P(i,j)}$, and $P(i,j)$ and $R(i,j)$ denote precision and recall, respectively, measured for class i and cluster j. The F-measure is a good index of cluster purity with high values indicating that each cluster contains objects predominantly from one class. Figure 2 summarizes the results obtained on all three data sets.

The standard Gaussian mixture model (GMM) has been evaluated in different settings, one of them using whole images (GMM-whole). For the caltech subsets where ground truth is available in the form of bounding boxes, we have also tested the method using only the image content within the ground truth foreground subwindows (GMM-WGT). The generic object detector of [1] provides for each image, the bounding box with the highest probability of corresponding to an object. The graphs show the result of applying GMM on the image content lying within these boxes as well. From this Figure we see that our approach greatly outperforms GMM using full images (with the segment selection procedure for foreground lo-

calization marginally outperforming subwindow discovery method). Furthermore, somewhat surprisingly, our approach also does much better than clustering applied to the foreground ground truth subwindows. We speculate that this is because the manual annotations are subjective and unreliable. Particularly in classes with high degree of variance, the human-selected boxes might work against the clustering attempt as the content expressed within the foreground regions of images within the same class might not be similar. On the other hand, unsurprisingly, the results of applying GMM-Obj are poor since determining objects from a single still image is an ill defined task

However, we chiefly compare against the results of the "foreground focus" (FF) method described in[17] and the multiple segmentations (Multi-Seg) method of [24]. Significantly, our system also outperforms the results reported in [17] despite their algorithm using a sophisticated semi-local representation encoding relative location of features in spatial neighborhoods. The difference in performance is especially noticeable on the most challenging MSRC-v1 data set, which contains objects at different scales and in different positions within the image.

6.2 Foreground initialization

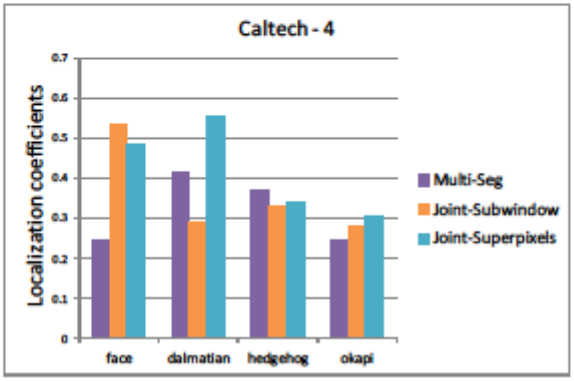
Dataset	Whole Image	Object Box	CoSeg Box
Caltech-4	0.685	0.705	**0.864**
Caltech-10	0.574	0.545	**0.72**
MSRC-v1	0.719	0.676	**0.877**

Table 1: FMeasure for different initialization methods

We have evaluated several different ways of initializing the foreground masks for the images. In particular, we report for 3 different methods: Initializing masks to be full images, initializing masks to be the most probable region in the image to contain an object as determined by a class generic object detector [1] and finally initializing foregrounds using the pairwise image cosegmentation method described in previous section. A summary of the results in terms of clustering quality is given in Table 6.2. From the table it is clear that the cosegementation approach, despite the high cost provides the best results consistently.

6.3 Foreground Localization

We now proceed to evaluate our approach in terms of object localization accuracy. In [17], the authors determine the quality of the foreground localization by examining the normalized sum of the weights inside the ground truth foreground. While their performance on this metric does indicate that the foreground features



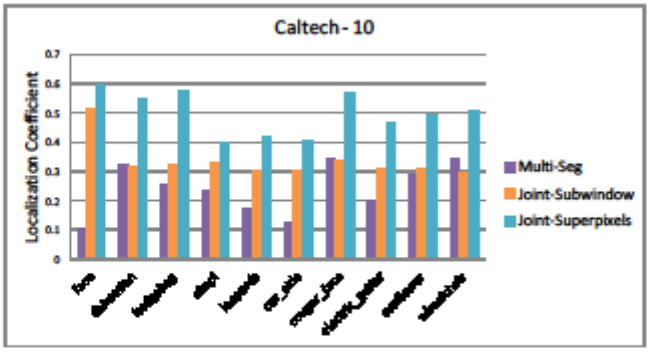


Fig. 3: Average localization scores achieved by our methods on all images from each ground truth class in the 4-class and the 10-class subsets of Caltech101. We also show the localization scores achieved by Multi-Seg. Please see text for more details.

get higher weight than background features, there is no clear way of determining the actual locality and extent of the foregrounds in the images. Furthermore, with their metric, it is possible to get a high score by having just a few very highly weighted foreground features. Instead, it is useful for many applications to determine the actual location and size of the foreground. Our algorithm generates a natural solution to this requirement in the form of bounding boxes for the

foreground, when the localization method is subwindow discovery, and foreground segments [2], when the localization method is segment selection. We measure the quality of the foreground localization by using a metric commonly used in object detection: $J_n = area(x_n \bigcap x_n^{GT})/area(x_n \bigcup x_n^{GT})$ where x_n^{GT} is the ground truth for the object in image n. For subwindow discovery localization method, we use the bounding box ground truth provided for the images in Caltech 101 and for segment selection method, we use the full object contour ground truth provided. Figure 3 shows the localization scores achieved with our method on all images of the 4-class and the 10-class subsets of Caltech101. We also include the localization scores achieved by [24]. It is clear that the foreground localization by the segment selection method is superior to subwindow discovery, however, it does not make a significant difference in terms of F-measure scores. We speculate that this may be due to the restricted nature of the dataset with highly correlated background content appearing in the discovered subwindows which aid in clustering foregrounds correctly. While studying the scores, we want to emphasize that these are calculated with respect to the manually annotated ground truth. As we have already seen in the case of bounding boxes, they are somewhat arbitrary. In our method, foreground detection is optimized for image clustering. So it is reasonable to get foreground which are inconsistent with the ground truth, but nevertheless play a role in improving image clustering.

A brief note on our interpretation of the methods in [24]: We point out that this method was designed for a different task: it does not explicitly cluster the images or specify which segments are foregrounds. Nevertheless, we tried adapting this method to work on our task in two different ways: (a) We ran the code of [24]: for each image I, multiple segmentations were computed and a topic model was fit to the segments. Cluster membership was determined as the topic (T_I) of the segment (S_{best}) with the smallest KL divergence to its topic. Then, to localize the foreground, we selected all segments having T_I as the most probable topic from the segmentation containing S_{best}. (b) We used the super-pixels of our method as input to [24] and then applied the procedure described in (a) for clustering and localization (we also tried using the most frequently occurring topic as cluster membership criterion, with no improvement in performance). We have included the results for (a) in the plots in Figures 2 and 3. The results for (b) are very similar. In short, both the cases yielded much lower accuracy than our approach.

Finally, our algorithms are quite fast thanks to the very efficient foreground localization methods. For instance, on Caltech 4 subset, the EM approach based on branch-and-bound completes all its iterations in 300 seconds whereas the segment-selection method runs in 40 seconds. Figure 4 shows some examples of foreground prediction for our method both in terms of discovered subwindows and selected bottom-up segments. Please refer to supplemental data for more visualizations.

[2] super pixels with high foreground scores. We deem a superpixel, s_n^i to be part of the foreground at the end of the EM run if $\pi_n^i > 0.3$

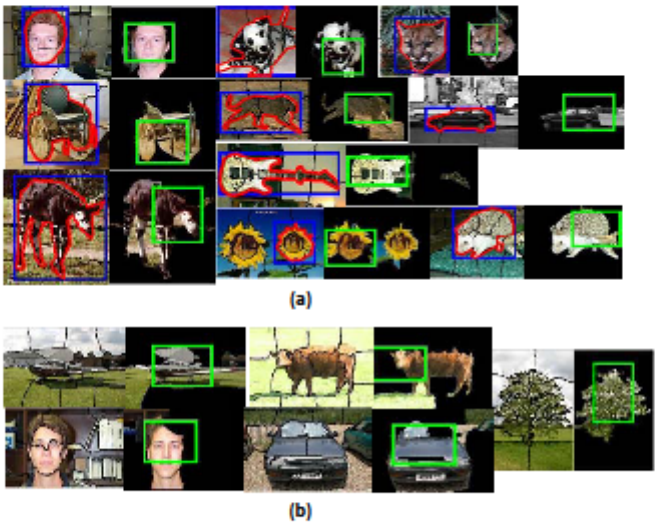


Fig. 4: (a): Examples of foreground prediction in images from the 10-class subset of Caltech101. Image on the left of each pair shows the super pixels obtained through bottom-up segmentation. The box in blue and the contour in red are the ground truth for the object location in the image. The image on the right of each pair shows the foreground discovered as a collection of super pixels (selected if $\pi_n^1 > 0.3$) by the segment selection method of localization. The box in green is the foreground extent predicted by subwindow discovery method. (b): Sample results for MSRC-v1 dataset.

7 Conclusions

Unsupervised foreground discovery is an important but difficult means of extracting structure from large unlabeled image datasets. In this work, we have developed a probabilistic method to perform simultaneous image clustering and foreground localization in unlabeled collections. We have shown that harnessing the natural synergy between the two tasks leads to improved performance at both the tasks. In the process, we have formulated two novel methods for discovering and representing object foregrounds by associating and efficiently estimating latent variables corresponding to bounding boxes and image segment sets. Our method can efficiently localize object foregrounds without resorting to expensive sliding window mechanisms. We note that our assumption that each image contains one of K objects and the simplicity of our appearance model allow us to cast foreground

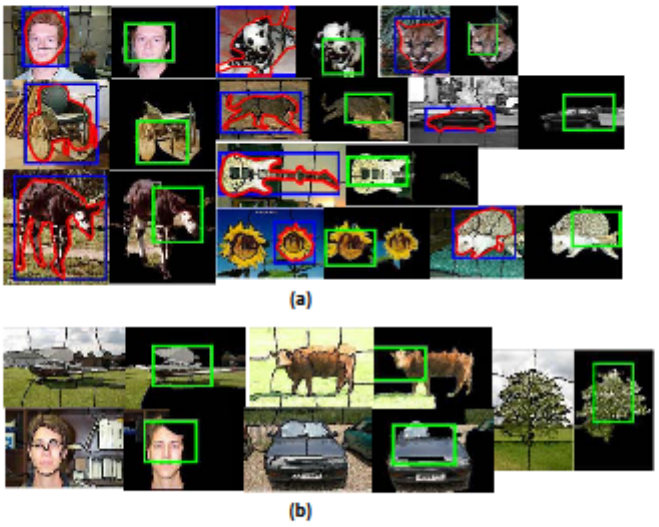


Fig. 4: (a): Examples of foreground prediction in images from the 10-class subset of Caltech101. Image on the left of each pair shows the super pixels obtained through bottom-up segmentation. The box in blue and the contour in red are the ground truth for the object location in the image. The image on the right of each pair shows the foreground discovered as a collection of super pixels (selected if $x_n^1 > 0.3$) by the segment selection method of localization. The box in green is the foreground extent predicted by subwindow discovery method. (b): Sample results for MSRC-v1 dataset.

7 Conclusions

Unsupervised foreground discovery is an important but difficult means of extracting structure from large unlabeled image datasets. In this work, we have developed a probabilistic method to perform simultaneous image clustering and foreground localization in unlabeled collections. We have shown that harnessing the natural synergy between the two tasks leads to improved performance at both the tasks. In the process, we have formulated two novel methods for discovering and representing object foregrounds by associating and efficiently estimating latent variables corresponding to bounding boxes and image segment sets. Our method can efficiently localize object foregrounds without resorting to expensive sliding window mechanisms. We note that our assumption that each image contains one of K objects and the simplicity of our appearance model allow us to cast foreground

clustering and localization elegantly as a single *joint* optimization, something has never been done until now. Furthermore, we empirically show that the approach outperforms methods that make more complex assumptions but that then have to resort to alternation between distinct objectives (e.g., [17]) or to a two-step solution (e.g., [24]) to solve the problem. We believe there is high value in simple models shown to perform well in practice. In the future we are interested in extending the work to videos where the task is a natural fit. Our probabilistic formulation also enables straightforward integration of non-visual cues such as text or tags associated to the images, which may yield more semantically meaningful clusters. The software implementing our algorithm will be made available upon publication.

References

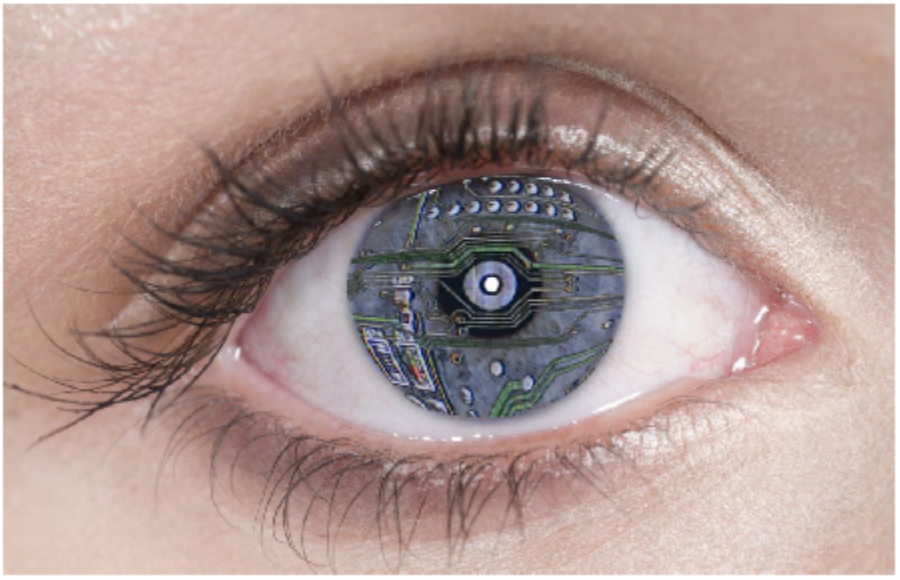
1. Alexe, B., Deselaers, T., Ferrari, V.: What is an object? In: CVPR (2012)
2. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. NIPS (2003)
3. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: SODA (2007)
4. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. Journal of Machine Learning Research. **3** (2003)
5. Borenstein, E., Sharon, E., Ullman, S.: Combining topdown and bottom-up segmentation. In: CVPR Workshop on Perceptual Organization in Computer Vision (2004)
6. Chen, Y., Wang, J.: Image categorization by learning and reasoning with regions. Journal of Machine Learning Research. **5**, 913939 (2004)
7. Chum, O., Zisserman, A.: An exemplar model for learning object classes. In: CVPR (2007)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
9. Deselaers, T., Alexe, B., Ferrari, V.: Localizing objects while learning their appearance. In: ECCV (2010)
10. Duygulu, P., Barnard, K., deFreitas, N., Forsyth, D.: Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In: ECCV (2002)
11. Fergus, R., Fei-Fei, L., Perona, P., Zisserman, A.: Learning object categories from googles image search. In: ICCV (2005)
12. Fritz, M., Schiele, B.: Decomposition, discovery and detection of visual categories using topic models. In: CVPR (2008)
13. Griffiths, T., Steyvers, M.: Finding scientific topics. PNAS. **101**, 5228–5235 (2004)
14. Itti, L., Koch, C.: Computational modelling of visual attention. Nature Reviews Neuroscience **2**(3), 194–203 (2001)
15. Kim, G., Torralba, A.: Unsupervised detection of regions of interest using iterative link analysis (2009)
16. Lampert, C., Blaschko, M., Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. In: CVPR (2008)
17. Lee, Y., Grauman, K.: Foreground focus: Unsupervised learning from partially matching images. IJCV. **85**, 143–166 (2009)
18. Leibe, B., Schiele, B.: Interleaved object categorization and segmentation. In: BMVC (2003)
19. Liu, C., Yuen, J., Torralba, A.: Sift flow: dense correspondence across different scenes and its applications. PAMI (2010)
20. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV. pp. 91–110 (2004)
21. Marr, D.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. New York: Freeman (1982)
22. Nguyen, M., Torresani, L., de la Torre, F., Rother, C.: Weakly supervised discriminative localization and classification: a joint learning process. In: ICCV (2009)

23. Rother C.and Minka, T., Blake, A., Kolmogorov, V.: Cosegmentation of image pairs by histogram matching - incorporating a global constraint into mrfs. In: CVPR, pp. 993–1000 (2006)
24. Russell, B., Efros, A., Sivic, J., Freeman, W.T., Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: CVPR (2006)
25. Shi, J.: Normalized cuts segmentation code. URL http://www.cis.upenn.edu/ jshi/software/
26. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering objects and their location in images. In: CVPR, pp. 370–377 (2005)
27. Sudderth, E., Torralba, A., Freeman, W., Willsky, A.: Learning hierarchical models of scenes, objects, and parts. In: ICCV (2005)
28. Tu, Z., Chen, X., Yuille, A., Zhu, S.: Image parsing: unifying segmentation, detection and recognition. IJCV. p. 113140 (2005)
29. Tuytelaars, T., Lampert, C., Blaschko, M., Buntine, W.: Unsupervised object discovery: A comparison. IJCV. pp. 284–302 (2010)
30. Yu, S., Shi, Y.: Object-specific figure-ground segregation. In: CVPR (2003)

Cerebrum, January 2011

How Brains Are Built: Principles of Computational Neuroscience

By Richard Granger, Ph.D.



Bernhard Lang/Photographer's Choice/Getty Images

Editor's note: The goal of computational neuroscience is to understand the brain and its mechanisms well enough to artificially simulate their functions. In some areas, like hearing, vision, and prosthetics, there have been great advances in the field. Yet there is still much about the brain that is unknown and therefore cannot be artificially replicated: How does the brain use language, make complex associations, or organize learned experiences? Once the neural pathways responsible for these and many other functions are fully understood and reconstructed, we will have the ability to build systems that can match—and maybe even exceed—the brain's capabilities.

Article available online at http://dana.org/news/cerebrum/detail.aspx?id=30356

"If I cannot build it, I do not understand it." So said Nobel laureate Richard Feynman, and by his metric, we understand a bit about physics, less about chemistry, and almost nothing about biology.[1]

When we fully understand a phenomenon, we can specify its entire sequence of events, causes, and effects so completely that it is possible to fully simulate it, with all its internal mechanisms intact. Achieving that level of understanding is rare. It is commensurate with constructing a full design for a machine that could serve as a stand-in for the thing being studied. To understand a phenomenon sufficiently to fully simulate it is to understand it *computationally*.

"Computation" does not refer to computers per se; rather it refers to the underlying principles and methods that make them work. As Turing Award recipient Edsger Dijkstra said, computational science "is no more about computers than astronomy is about telescopes."[2] Computational science is the study of the hidden rules underlying complex phenomena from physics to psychology.

Computational neuroscience, then, has the aim of understanding brains sufficiently well to be able to simulate their functions, thereby subsuming the twin goals of science and engineering: deeply understanding the inner workings of our brains, and being able to construct simulacra of them. As simple robots today substitute for human physical abilities, in settings from factories to hospitals, so brain engineering will construct stand-ins for our mental abilities—and possibly even enable us to fix our brains when they break.

Brains and Their Construction

Brains, at one level, consist of ion channels, chemical pumps, specialized proteins. At another level, they contain several types of neurons connected via synaptic junctions. These are in turn composed into networks consisting of repeating modules of carefully arranged circuits. These networks are arrayed in interacting brain structures and systems, each with distinct internal wiring and each carrying out distinct functions. As in most complex systems, each level arises from those below it but is not readily reducible to its constituents. Our understanding of an organism depends on our understanding of its component organs, but also on the ongoing interactions among those parts, as is evident in differentiating a living organism from a dead one.

For instance, kidneys serve primarily to separate and excrete toxins from blood and to regulate chemical balances and blood pressure, so a kidney simulacrum would entail a nearly complete set of chemical and enzymatic reactions. A brain also monitors many critical regulatory mechanisms, and a complete understanding of it will include detailed chemical and biophysical characteristics.

But brains, alone among organs, produce thought, learning, recognition. No amount of engineering has yet equaled, let alone surpassed, brains' abilities at these tasks. Despite huge efforts and

large budgets, we have no artificial systems that rival humans at recognizing faces, nor understanding natural languages, nor learning from experience.

There are, then, crucial principles that brains encode that have so far eluded the best efforts of scientists and engineers to decode. Much of computational neuroscience is aimed directly at attempting to decipher these principles.

Today we cannot yet fully simulate every aspect of a kidney, but we have passed a decisive threshold: we can build systems that replicate kidney principles so closely that they can supplant their function in patients who have suffered kidney loss or damage. Artificial kidneys do not use the same substrate as real kidneys; circuits and microfluidics take the place of cells and tissue, yet they carry out operations that are equivalent, and lifesaving, to the human bodies that use them. A primary long-term goal of computational neuroscience is to derive scientific principles of brain operation that will catalyze the comparable development of prosthetic brains and brain parts.

Do We Know Enough About Brains to Build Them?

As with any complex system, in the absence of full computational understanding of the brain, we proceed by collecting constraints: experimentally observable data can rule out potential explanations. The more we can rule out, the closer we are to hypotheses that can account for the facts. Many constraining observations have usefully narrowed our understanding of how mental activity arises from brain circuitry; these can be organized into five key categories.

Brain component allometry: Remarkably tight relationships hold between a brain's overall size and the size of its constituent components. Just knowing the overall brain size of any mammal, we can with great precision predict the size of all component structures within the brain. Thus, with few exceptions, brains apparently do not and cannot choose which structures to differentially expand or reconfigure.[3–11] So, quite surprisingly, rather than a range of different circuits, or even selective resizing of brain components, human brains are instead largely built from the same components as other mammalian brains, in the same circuit layouts, with highly predictable relative sizes. Apparently a quantitative change (brain size) results in a qualitative one (uniquely human computational capabilities).[9, 12]

Telencephalic uniformity: Circuits throughout the forebrain (telencephalon) exhibit notably similar repeated designs,[6,13] with few exceptions,[14–19] including some slightly different cell types, circuit structures, and genes. Yet brain areas purported to underlie unique human abilities (e.g., language) barely differ from other structures; there are no extant hypotheses of how the modest observed genetic or anatomical differences could engender exceedingly different functions. Taken together, these findings

intimate the existence of a few elemental core computational functions that are re-used for a broad range of apparently different sensory and cognitive operations.

Anatomical and physiological imprecision: Evidence suggests that neural components are surprisingly sloppy (probabilistic) in their operation, very sparsely connected, low-precision, and extraordinarily slow,[20–22] despite exhibiting careful timing under some experimental conditions.[23–27] Either brains are far more precise than we yet understand, or else they carry out families of algorithms whereby precise computations arise from imprecise components.[28–31] If so, this greatly constrains the types of operations that any brain circuits could be engaged in.

Task specification: Though artificial telephone operators field phone inquiries with impressive voice recognition, we know that they could do far better. The only reason we know this is that human operators substantially outperform them; there are no other formal specifications whatsoever that characterize the voice recognition task.[32,33] Engineers began by believing that they understood the task sufficiently to construct artificial operators. It has turned out that their specification of the task does not match the actual, still highly elusive set of steps that humans actually perform in recognizing speech. Without formal task specifications, the only way to equal human performance may be to come to understand the brain mechanisms that give rise to the behavior.

Parallel processing: Some recognition tasks take barely a few hundred milliseconds,[34,35] corresponding to no more than hundreds of serial neural steps (of milliseconds each), strongly indicating myriad neurons acting in parallel,[36] imposing a very strong constraint on the types of operations that individual neurons could be carrying out. Yet parallelism in computer science, even on a small scale, such as two or three simultaneous operations, has proven very elusive. Why, for instance, don't our dual-core or quad-core computers run two or four times faster than single-core systems? The (painfully direct) answer is that we simply do not yet know how to divide most software into parts that can effectively exploit the presence of these additional hardware elements. Even for readily parallelizable software, it is challenging to design hardware that yields scalable returns as processors are added.[37,38] It is increasingly possible that principles of brain architecture may help identify novel and powerful parallel machine designs.

From Circuits to Algorithms to Prosthetics

There are several promising instances in which different laboratories (even laboratories that are competing with each other) have arrived at substantial points of agreement about what certain brain areas are likely doing. A notable success story arises from studies of the basal ganglia, which takes two kinds of

inputs: sensory information from the neocortex, and "reward" and "punishment" information arising from external stimuli. We are close to computationally understanding this large chunk of the brain, which apparently carries out just one of our primary learning abilities: our slow "trial and error" learning (studied in computational neuroscience as "reinforcement learning"), underlying our ability to acquire such skills as riding a bike.[30, 39–65]

In addition, there is a growing consensus that circuits in the neocortex, by far the largest set of brain structures in humans, carry out another, quite different kind of learning: the ability to rapidly learn new facts and to organize newly acquired knowledge into vast hierarchical structures that encode complex relationships, such as categories and subcategories, episodes, and relations.[28, 66–74]

And these two systems are connected to each other, via far-reaching cortico-basal ganglia (aka cortico-striatal) loops .[49] The basal ganglia system carries out the computational operations of skill learning (reinforcement learning) while cortical circuits computationally construct vast hierarchies of facts and relations among facts. Interestingly, computational research on reinforcement learning has found that adding hierarchies to the process can greatly improve learning performance .[75,76] Our ancestors (reptiles and early mammals) were largely driven by the basal ganglia, whereas mammalian evolution has hugely expanded the relative size of the neocortex. By consistently increasing the size ratio of the neocortex to the basal ganglia, mammalian brain evolution may be solving a specific computational puzzle.[29, 75–79] Our understanding of human and animal learning abilities is being advanced by these computational studies, and we are developing novel methods for machine learning, enabling more powerful computer algorithms for analysis of complex data ranging from medical to commercial to financial applications.

Meanwhile, as study of these primary cortico-striatal brain structures remains very much still in progress, great advances have been made in deep, computational understanding of certain circumscribed brain systems, in particular those involved in early sensory transduction and perception. The results have been striking.

Analysis of cochlear mechanisms has led to the construction of prosthetics that serve today as cures for more than 100,000 people who have lost their hearing.[80] Retinal prosthetics are in advanced development.[81–85] In a recent study, patients with retinal implants recognized printed letters of size and distance comparable to reading a book in relatively low light. And experimental prosthetic arms can respond to brain-initiated control; people learn to control the arm simply by deciding to move it.[86, 87]

These sensory and motor findings have also led to formalizations of the general problem of acting in environments that are only partly observable and are dynamically changing, such as robotics or automated navigation; the result is a set of increasingly impressive robotic methods that see and navigate in complex surroundings.[88] In a series of trials run by the Department of Defense over the last several

years, vehicles were, for the first time, able to navigate through real urban traffic, merging, passing, parking, and negotiating intersections, with no human control. Retinal algorithms operate equally well on other sensors such as radar; and prosthetic limb algorithms are wholly applicable to robots. Many of the algorithms that operate robots and automated vehicles are closely related to those that operate prosthetic limbs.

As we come to computationally understand how these peripheral sensorimotor systems work, the distinction between natural and artificial is being eroded. A breed of robots that share many of our own dexterity and perceptual abilities is likely to emerge directly from this research. As these increasingly biologically-based robots, or biots, come to replace human skilled labor, the economic and social consequences may be substantial.

From Percept to Concept

The primary differences between human brains and those of other animals lie not in our sensory or motor mechanisms, which are largely shared across many species, but rather in cognitive abilities: association, representation, reasoning. Despite great advances in peripheral prosthetics, there is no commensurate understanding of advanced cognition.

The abilities of peripheral circuits (retina, cochlea, initial thalamic and cortical regions) are largely built in at birth via genetic programs and shaped in early childhood during developmentally critical periods. In contrast, the rest of the neocortex will use those built-in systems to acquire masses of specific information about the environment over a lifetime. Neocortical circuits are not born with knowledge of particular scenes, faces, or actions; these are acquired through sensorimotor experience: observing and interacting with objects and events in our surroundings. Cortical circuits are engaged almost entirely in fact learning: rapid, permanent acquisition and organization of everyday occurrences. The low-level biological mechanisms underpinning long-term fact learning (permanent, anatomical synaptic changes, rather than inherently ephemeral chemical changes) are becoming understood.[89] But the neocortex is not just a passive warehouse of billions of isolated facts; we can arbitrarily associate them, recall them, embellish them.[33] Association, recall, retrieval, organization—all that we can actually do with memory—depends on mechanisms that are as yet still unknown.

Early cortical areas, then, deal with recognizing objects (even in different lighting, settings, and clutter), but some laboratories are increasingly focusing on cortical circuits that are beyond the early sensory areas: the vast remainder of the neocortex that somehow encodes sequences, associations, and abstract relations.[33, 90–99]

Seeing a phone, we perceive not only its visual form but also its affordances (calling, texting, photographing, playing music), our memories of it (when we got it, where we have recently used it), and a

years, vehicles were, for the first time, able to navigate through real urban traffic, merging, passing, parking, and negotiating intersections, with no human control. Retinal algorithms operate equally well on other sensors such as radar; and prosthetic limb algorithms are wholly applicable to robots. Many of the algorithms that operate robots and automated vehicles are closely related to those that operate prosthetic limbs.

As we come to computationally understand how these peripheral sensorimotor systems work, the distinction between natural and artificial is being eroded. A breed of robots that share many of our own dexterity and perceptual abilities is likely to emerge directly from this research. As these increasingly biologically-based robots, or biots, come to replace human skilled labor, the economic and social consequences may be substantial.

From Percept to Concept

The primary differences between human brains and those of other animals lie not in our sensory or motor mechanisms, which are largely shared across many species, but rather in cognitive abilities: association, representation, reasoning. Despite great advances in peripheral prosthetics, there is no commensurate understanding of advanced cognition.

The abilities of peripheral circuits (retina, cochlea, initial thalamic and cortical regions) are largely built in at birth via genetic programs and shaped in early childhood during developmentally critical periods. In contrast, the rest of the neocortex will use those built-in systems to acquire masses of specific information about the environment over a lifetime. Neocortical circuits are not born with knowledge of particular scenes, faces, or actions; these are acquired through sensorimotor experience: observing and interacting with objects and events in our surroundings. Cortical circuits are engaged almost entirely in fact learning: rapid, permanent acquisition and organization of everyday occurrences. The low-level biological mechanisms underpinning long-term fact learning (permanent, anatomical synaptic changes, rather than inherently ephemeral chemical changes) are becoming understood.[89] But the neocortex is not just a passive warehouse of billions of isolated facts; we can arbitrarily associate them, recall them, embellish them.[33] Association, recall, retrieval, organization—all that we can actually do with memory—depends on mechanisms that are as yet still unknown.

Early cortical areas, then, deal with recognizing objects (even in different lighting, settings, and clutter), but some laboratories are increasingly focusing on cortical circuits that are beyond the early sensory areas: the vast remainder of the neocortex that somehow encodes sequences, associations, and abstract relations.[33, 90–99]

Seeing a phone, we perceive not only its visual form but also its affordances (calling, texting, photographing, playing music), our memories of it (when we got it, where we have recently used it), and a

wealth of potential associations (our ringtone, whom we might call, whether it is charged, etc.). The questions of how cross-modal information is learned and integrated, and in what form the knowledge is stored—how percepts become concepts—now constitute the primary frontier of work in computational neuroscience. In this borderland between perception and cognition, the peripheral language of the senses is transmuted to the internal lingua franca of the brain, freed from literal sensation and formulated into internal representations that can include a wealth of associations.

Even our simplest perceptions often rely on top-down processing: using stored memory representations to inform our ongoing perception and recognition. In some circumstances, we can recognize objects in just tens of milliseconds,[34,35] so rapidly that it is unlikely that any top-down pathways are yet engaged. Yet once we're beyond simple recognition, to the far richer range of inference, association, and even language, memories strongly influence our perceptions. Merely thinking of a car is sufficient to activate the same early visual areas that would have been triggered by actually seeing the car, including its shape, size, color, and other features.[100–102]

These early visual areas are just one instance of the spread of activation from a triggering memory.[103–105] Thinking of a car may also activate many other areas, as yet largely unmapped, that encode knowledge of how to open car doors, turn ignition keys, steer, accelerate, brake—or information about what particular car you own, where it is parked, and so on. Today we can experimentally test for visual shape information because we know a great deal about how to decode neural responses that occur in early visual areas,[106] but we have comparatively modest data for other associative knowledge.[107–109] Computational models of spreading activation[110,111] are now striving to make contact with specific neural mechanisms and brain pathways, to arrive at convergent hypotheses like those of peripheral sensory systems.

Computing Individual Differences: From Neurotypes to Cognotypes

Though all of us have extraordinarily similar brains, even small differences can be striking. Whether particular characteristics are genetic, developmental, or learned is still often impossible to ascertain, but individual behavioral differences are highly likely to directly correspond to individual brain differences, whether genetic or acquired. Most work in computational neuroscience—from perception to cognition, from anatomy to computational models—has focused on one agent at a time, one brain at a time. A further frontier will be to confront differences among individuals.

Our bodies are built by genetic programs that became locked into particular patterns early on in mammalian evolution: four appendages; eyes above nose above mouth between ears; ten fingers and ten toes. We are not optimized to have just these features and no others; most of the variations that we might imagine—nose above eyes; five limbs; tentacles instead of hands—have never been tried by evolution,

patient. And there are risks: the surgical implantation procedure may lead to a higher incidence of meningitis.[116,117] Moreover, there are social complications: some in the deaf community find cochlear implants to be ethically misplaced, arguing that the deaf should not be thought of as disabled at all, but rather as a "minority cultural group."[118]

What of brain parts that are deeper than just the peripheral hearing system? Traumatic brain injury can cause debilitating deficits in memory and cognition; at present, such injuries are extremely difficult even to diagnose, let alone to treat. Implants to restore lost cognitive abilities for such accident victims would be revolutionary, and would be welcomed.

But if implants existed for accident-induced cognitive losses, could they also be used to augment uninjured cognitive function? There is suggestive evidence from drugs: some Alzheimer's medications may improve memory in people with mild cognitive impairment—but the FDA has not yet approved the use of any treatments for these lesser conditions.[119,120] How would regulators at the FDA react if it became possible to augment our brains—implants to help us think faster or to increase our memory capacity? The economic, social, and political concomitants of such technology would surely eclipse those arising from cochlear implants.

Each brain contains idiosyncrasies; our brains define who we are. The way we interact, the kinds of decisions we make, the connections we perceive—all arise from the still-obscure mechanisms of the vast span of thalamocortical circuits and cortico-striatal loops in our heads. These repeating components give us our mammalian abilities, our uniquely human faculties, and our individual characteristics. The computational understanding of individual and group differences will likely lead to a new science of different types of cognitive behavior, with implications ranging from law to education. The formerly familiar terrain of human nature may appear quite different in this light; perhaps, arriving there, we will truly know the place for the first time.

Our abilities are not inimitable; brain circuits are circuits, albeit nonstandard ones, and they will yield to analysis. As computational neuroscience comes to demystify them, we verge on an era of new frontiers in science and medicine, in which we can increasingly repair, enhance, and likely supplant the biological engines we think with.

Richard Granger, Ph.D., is a professor at Dartmouth with faculty positions in the departments of psychological and brain sciences, computer science, and the Thayer School of Engineering. He directs Dartmouth's interdisciplinary Brain Engineering Laboratory, with research projects ranging from computation and robotics to neuroimaging and cognitive neuroscience. He has authored more than 100 scientific papers and holds numerous issued patents, is an elected fellow of the American Association for the Advancement of Science (AAAS), and serves on the boards of a number of technology corporations and government agencies. He is co-inventor of FDA-approved devices and drugs in clinical trials, and has been the principal architect of a series of advanced computational systems for military, commercial, and medical applications.

References

1. Feynman, R. In Hawking, S. (2001). The universe in a nutshell (p. 83). Bantam.
2. Dijkstra, E. (2001). Denken als Discipline (Discipline in Thought) [Video interview]. Retrieved from http://www.cs.utexas.edu/users/EWD/video-audio/NoorderlichtVideo.html.
3. Jerison, H. (1973). Evolution of the brain and intelligence. Academic Press.
4. Finlay, B., Innocenti, G., & Scheich, H. (1991). The neocortex: Ontogeny and phylogeny. Plenum Press.
5. Finlay, B., & Darlington, R. (1995). Linked regularities in the development and evolution of mammalian brains. Science, 268, 1578–1584.
6. Striedter, G. F. (2005). Principles of brain evolution. Sinauer Associates.
7. Falk, D., & Gibson, K. (2001). Evolutionary anatomy of the primate cerebral cortex. Cambridge University Press.
8. Sherwood, C., Holloway, R., Semendeferi, K., & Hof, P. (2010). Inhibitory interneurons of the human prefrontal cortex display conserved evolution of the phenotype and related genes. Proceedings of the Royal Academy of Science B, 277, 1011–1020.
9. Lynch, G., & Granger, R. (2008). Big brain. Palgrave Macmillan.
10. Herculano-Houzel, S. (2009). The human brain in numbers: A linearly scaled-up primate brain. Frontiers in Neuroscience, 3, 1–11.
11. Semendeferi, K., Teffer, K., Buxhoeveden, D., Park, M., Bludau, S., Amunts, K., . . . Buckwalter, J. (2010). Spatial organization of neurons in the prefrontal cortex sets humans apart from great apes. Cerebral Cortex. doi: 10.1093/cercor/bhq191.
12. Amati, D., & Shallice, T. (2007). On the emergence of modern humans. Cognition, 103(3), 358–385.

13. Jones, E. G., & Rakic, P. (2010). Radial columns in cortical architecture: It is the composition that counts. Cerebral Cortex, 20(10), 2261–2264.

14. Nimchinsky, E., Glissen, E., Allman, J., Perl, D., Erwin, J., & Hof, P. (1999). A neuronal morphologic type unique to humans and great apes. Proceedings of the National Academy of Science, 96, 5268–5273.

15. Galuske, R., Schlote, W., Bratzke, H., & Singer, W. (2000). Interhemispheric asymmetries of the modular structure in humans. Science, 289, 1946–1949.

16. Buxhoeveden, D., Switala, A., Roy, E., Litaker, M., & Casanova, M. (2001). Morphological differences between minicolumns in human and nonhuman primate cortex. American Journal of Physical Anthropology, 115, 361–371.

17. Lai, C., Fisher, S., Hurst, J., Levy, E., Hodgson, S., Fox, M., . . . Monaco, A. (2000). The SPCH1 region on human 7q31: Genomic characterization of the critical interval and localization of translocations associated with speech and language disorder. American Journal of Human Genetics, 67, 357–368.

18. Evans, P., Gilbert, S., Mekel-Bobrov, N., Vallender, E., Anderson, J., Vaez-Azizi, L., . . . Lahn, B. (2005). Microcephalin, a gene regulating brain size, continues to evolve adaptively in humans. Science, 309, 1717–1720.

19. Mekel-Bobrov, N., Gilbert, S., Evans, P., Vallender, E., Anderson, J., Hudson, R., . . . Lahn, B. (2005). Ongoing adaptive evolution of ASPM, a brain size determinant in Homo sapiens. Science, 309, 1720.

20. Braitenberg, V., & Schüz, A. (1998). Cortex: Statistics and geometry of neuronal connectivity. Springer-Verlag.

21. Häusser, M., & Mel, B. (2003). Dendrites: Bug or feature? Current Opinion in Neurobiology, 13, 372–383.

22. Fuhrmann, G., Segev, I., Markram, H., & Tsodyks, M. (2002). Coding of temporal information by activity-dependent synapses. Journal of Neurophysiology, 87, 140–148.

23. Singer, W. (1999). Neuronal synchrony: A versatile code for the definition of relations? Neuron, 24(1), 49–65, 111–125.

24. Singer, W. (2010). Distributed processing and temporal codes in neuronal networks. Cognitive Neurodynamics 3, 189–196.

25. Traub, R., Bibbig, A., LeBeau, F., Buhl, E., & Whittington, M. (2004). Cellular mechanisms of neuronal population oscillations in the hippocampus in vitro. Annual Review of Neuroscience, 27, 247–278.

26. Wang, X. (2010). Neurophysiological and computational principles of cortical rhythms in cognition. Physiological Reviews, 90(3), 1195–1268.

27. Clopath, C., Busing, L., Vasilaki, E., & Gerstner, W. (2010). Connectivity reflects coding: A model of voltage-based STDP with homeostasis. Nature Neuroscience, 13, 344–352.

28. Rodriguez, A., Whitson, J., & Granger, R. (2004). Derivation and analysis of basic computational operations of thalamocortical circuits. Journal of Cognitive Neuroscience, 16, 856–877.

29. Granger, R. (2005). Brain circuit implementation: High-precision computation from low-precision components. In Berger & Glanzman (Eds.), Replacement parts for the brain (pp. 277–294). MIT Press.

30. Granger, R. (2006). Engines of the brain: The computational instruction set of human cognition. AI Magazine, 27, 15–32.

31. Felch, A., & Granger, R. (2008). The hypergeometric connectivity hypothesis: Divergent performance of brain circuits with different synaptic connectivity distributions. Brain Research, 1202, 3–13.

32. Edelman, S. (1999). Representation and recognition in vision. MIT Press.

33. Edelman, S., & Intrator, N. (2003). Towards structural systematicity in distributed, statically bound visual representations. Cognitive Science, 27, 73–110.

34. Thorpe, S., Fize, D., & Marlot, C. (1996). Speed of processing in the human visual system. Nature, 381(6582), 520–522.

35. Stanford, T., Shankar, S., Massoglia, D., Costello, M., & Salinas, E. (2010). Perceptual decision making in less than 30 milliseconds. Nature Neuroscience, 13(3), 379–385.

36. Feldman, J., & Ballard, D. (1982). Connectionist models and their properties. Cognitive Science, 6, 205–254.

37. Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., . . . Yelick, K. (2009). A view of the parallel computing landscape. Communications of the ACM, 52, 56–67.

38. Moorkanikara, J., Felch, A., Chandrashekar, A., Dutt, N., Granger, R., Nicolau, A., & Veidenbaum, A. (2009). Brain-derived vision algorithm on high-performance architectures. International Journal of Parallel Programming, 37, 345–369.

39. Schultz, W., Dayan, P., & Montague, R. (1997). A neural substrate of prediction and reward. Science, 175, 1593–1599.

40. Schultz, W., Apicella, P., & Ljungberg, T. (1993). Responses of monkey dopamine neurons to reward and conditioned stimuli during a delayed response task. Journal of Neuroscience, 13, 900–913.

41. Schultz, W. (1998). Predictive reward signal of dopamine neurons. Journal of Neurophysiology, 80, 1–27.

42. Schultz, W. (2002). Getting formal with dopamine and reward. Neuron, 36, 241–263.

43. Suri, R., & Schultz, W. (2001). Temporal difference model reproduces anticipatory neural activity. Neural Computation, 13(4), 841–862.

44. Suri, R. (2001). Anticipatory responses of dopamine neurons and cortical neurons reproduced by internal model. Experimental Brain Research, 140(2), 234–240.

45. Sutton, R. S., & Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In Gabriel & Moore (Eds.), Learning and computational neuroscience: Foundations of adaptive networks,(pp. 497–537). MIT Press.

46. Sutton, R., & Barto, A. (1998). Reinforcement learning: An introduction. MIT Press.

47. Strick, P., Dum, R., & Mushiake, H. (1995). Basal ganglia loops with the cerebral cortex. In M. Kimura & A. Graybiel (Eds.), Functions of the cortico-basal ganglia loop (pp. 106–124). Springer-Verlag.

48. Gerfen, C., & Wilson, C. (1996). The basal ganglia. In Swanson, Bjorklund, & Hokfelt (Eds.), Handbook of Chemical Neuroanatomy, vol. 12 (pp. 371–468). Elsevier.

49. Alexander, G., & DeLong, M. (1985). Microstimulation of the primate neostriatum. I. Physiological properties of striatal microexcitable zones. Journal of Neurophysiology, 53, 14001–11416.

50. Graybiel, A., Aosaki, T., Flaherty, A., & Kimura, M. (1994). The basal ganglia and adaptive motor control. Science, 265, 1826–1831.

51. Graybiel, A. (1995). Building action repertoires. Current Opinion in Neurobiology, 5, 733–741.

52. Houk, J., Davis, J., & Beiser, D. (1995). Models of information processing in the basal ganglia. MIT Press.

53. Houk, J., & Wise, S. (1995). Distributed modular architectures linking basal ganglia, cerebellum, and cerebral cortex. Cerebral Cortex, 2, 95–110.

54. Knowlton, B., & Squire, L. (1993). The learning of categories: Parallel brain systems for item memory and category knowledge. Science, 262, 1747–1749.

55. Brucher, F. (2000). Reward-based learning and basal ganglia: A biologically realistic, computationally explicit theory. Unpublished doctoral dissertation, University of California.

56. Poldrack, R., Clark, J., Pare-Blagoev, E., Shohamy, D., Cresco Moyano, J., Myers, C., & Gluck, M. (2001). Interactive memory systems in the human brain. Nature, 414, 546–550.

57. Daw, N. (2003). Reinforcement learning models of the dopamine system and their behavioral implications. Unpublished doctoral dissertation, Carnegie Mellon University.

58. Frank, M. (2005). Dynamic dopamine modulation in the basal ganglia: A neurocomputational account of cognitive deficits in medicated and non-medicated Parkinsonism. Journal of Cognitive Neuroscience, 17, 51–72.

59. Laubach, M. (2005). Who's on first? What's on second? The time course of learning in corticostriatal systems. Trends in Neuroscience, 28, 509–511.

60. Yin, H., & Knowlton, B. (2006). The role of the basal ganglia in habit formation. Nature Reviews Neuroscience, 7(6), 464–476.

61. Swinehart, C., & Abbott, L. (2006). Dimensional reduction for reward-based learning. Network: Computation in Neural Systems, 17(3), 235–252.

62. Hazy, T., Frank, M., & O'Reilly, R. (2007). Towards an executive without a homunculus: Computational models of the prefrontal cortex/basal ganglia system. *Philosophical* Transactions of the Royal Society B, 362, 1601–1613.

63. Green, C., Pouget, A., & Bavelier, D. (2010). Improved probabilistic inference as a general learning mechanism with action video games. Current Biology, 20, 1573–1579.

64. Erickson, K., Boot, W., Basak, C., Neider, M., Prakash, R., Voss, M., Graybiel, A., . . . Kramer, A. (2010). Striatal volume predicts level of video game skill acquisition. Cerebral Cortex doi:10.1093/cercor/bhp293.

65. Samson, R., Frank, M., & Fellous, J. (2010). Computational models of reinforcement learning: The role of dopamine as a reward signal. Cognitive Neurodynamics, 4, 91–105.

66. Olshausen, B. (1996). Emergence of simple cell receptive field properties by learning a sparse code for natural images. Nature, 381, 607.

67. Douglas, R., & Martin, K. (2004). Neuronal circuits of the neocortex. Annual Review Neuroscience, 27, 419–451.

68. Friston, K. (2008). Hierarchical models in the brain. PLoS Computational Biology, 4, e1000211.

69. George, D., & Hawkins, J. (2009). Towards a mathematical theory of cortical microcircuits. PLoS Computational Biology, 5, e1000532.

70. Riesenhuber, M. (1999). Hierarchical models of object recognition in cortex. Nature Neuroscience, 2, 1019.

71. Lee, T., & Mumford, D. (2003). Hierarchical bayesian inference in the visual cortex. Journal of the Optical Society of America, 20, 1434–1448.

72. Granger, R., & Hearn, R. (2008). Models of the thalamocortical system. Scholarpedia, 2(11), 1796.

73. Nikolić, D., Häusler, S., Singer, W., & Maass, W. (2009). Distributed fading memory for stimulus properties in the primary visual cortex. PLoS Biology, 7(12), e1000260.

74. Smale, S., Rosasco, L., Bouvrie, J., Caponnetto, A., & Poggio, T. (2009). Mathematics of the neural response. Foundations of Computational Mathematics, 10(1), 67–91.

75. Dietterich, T. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research, 13, 227–303.

76. Barto, A., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. Discrete Event Dynamic Systems, 13(341–379).

77. Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112, 181–211.

78. Granger, R. (2006). The evolution of computation in brain circuitry. Behavioral and Brain Science 29, 17.

79. Barry, J., Kaelbling, L., & Lozano-Perez, T. (2010). Hierarchical solution of large Markov decision processes. Technical report, MIT.

80. NIDCD. (2009). Cochlear implants. Retrieved from www.nidcd.nih.gov/health/hearing/coch.asp

81. Weiland, J., Liu, W., & Humayun, M. (2005). Retinal prosthesis. Annual Review of Biomedical Engineering, 7, 361–401.

82. U.S. Dept. of Energy Office of Science. (2009). Artificial retina project. Retrieved from http://artificialretina.energy.gov

83. Chen, K., Yang, Z., Hoang, L., Weiland, J., Humayu, M., & Liu, W. (2010). An integrated 256-channel epiretinal prosthesis. IEEE Journal of Solid State Circuits, 45, 1946–1956.

84. Zhou, C., Tao, C., Chai, X., Sun, Y., & Ren, Q. (2010). Implantable imaging system for visual prosthesis. Artificial Organs, 34(6), 518–522.

85. Zrenner, E., Wilke, R., Bartz-Schmidt, K. U., Gekeler, F., Besch, D., Benav, H., Bruckmann, A., . . . Stett, A. (2009). Subretinal microelectrode arrays allow blind retinitis pigmentosa patients to recognize letters and combine them to words. 2nd International Conference on Biomedical Engineering and Informatics (pp. 1–4).

86. Moritz, C., Perlmutter, S., & Fetz, E. (2008). Direct control of paralysed muscles by cortical neurons. Nature, 456, 639–642.

87. Velliste, M., Perel, S., Spalding, M. C., Whitford, A. S., & Schwartz, A. B. (2008). Cortical control of a prosthetic arm for self-feeding. Nature, 453(7198), 1098–1101.

88. Thrun, S. (2000). Probabilistic algorithms in robots. AI Magazine, 21, 93–109.

89. Fedulov, V., Rex, C., Simmons, D., Palmer, L., Gall, C., & Lynch, G. (2007). Evidence that long-term potentiation occurs within individual hippocampal synapses during learning. Journal of Neuroscience, 27, 8031–8039.

90. Op de Beeck, H., Baker, C., DiCarlo, J., & Kanwisher, N. (2006). Discrimination training alters object representations in human extrastriate cortex. Journal of Neuroscience, 26, 13025–13036.

91. Li, N., & DeCarlo, J. (2008). Unsupervised natural experience rapidly alters invariant object representation in visual cortex. Science, 321, 1502–1507.

92. Wallisch, P., & Movshon, J. A. (2008). Structure and function come unglued in the visual cortex. Neuron, 60(2), 195–197.

93. Pinto, N., Cox, D., & DiCarlo, J. (2008). Why is real-world visual object recognition hard? PLoS Computational Biology, 4(1), e27.

94. Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. Annual Review of Neuroscience, 24, 1193–1216.

95. Cox, D., Meier, P., Oertelt, N., & DiCarlo, J. (2005). Breaking position invariant object recognition. Nature Neuroscience, 8, 1145–1147.

96. Geman, S. (2006). Invariance and selectivity in the ventral visual pathway. Journal of Physiology, 100, 212–224.

97. Yuille, A., & Kersten, D. (2006). Vision as Bayesian inference: Analysis by synthesis? Trends in Cognitive Sciences, 10(7), 301–308.

98. DiCarlo, J., & Cox, D. (2007). Untangling invariant object recognition. Trends in Cognitive Sciences, 11, 333–341.

99. Roy, J., Riesenhuber, M., Poggio, T., & Miller, E. (2010). Prefrontal cortex activity during flexible categorization. Journal of Neuroscience, 30, 8519–8528.

100. Kosslyn, S., Alpert, N., Thompson, W., Maljkovic, V., Weise, S., Chabris, C., . . . Buonanno, F. (1993). Visual mental imagery activates topographically organized visual cortex: PET Investigations. Journal of Cognitive Neuroscience, 5, 263–287.

101. Kosslyn, S., Thompson, W., Kim, I., & Alpert, N. (1995). Topographical representations of mental images in primary visual cortex. Nature, 378, 496–498.

102. Slotnick, S., Thompson, W., & Kosslyn, S. (2005). Visual mental imagery induces retinotopically organized activation of early visual areas. Cerebral Cortex, 15(10), 1570–1583.

103. Posner, M., & Snyder, C. (1975). Attention and cognitive control. In Solso (Ed.), Information processing and cognition: The Loyola symposium (pp. 55–85). Erlbaum.

104. Neely, J. (1977). Semantic priming and retrieval from lexical memory: Roles of inhibitionless spreading activation and limited-capacity attention. Journal of Experimental Psychology, 106, 226–254.

105. Ratcliff, R., & McKoon, G. (1978). Priming in item recognition: Evidence for the propositional structure of sentences. Journal of Verbal Learning and Verbal Behavior, 17, 403–417.

106. Kay, K., Naselaris, T., Prenger, R., & Gallant, J. (2008). Identifying natural images from human brain activity. Nature, 452, 352–355.

107. Naselaris, T., Prenger, R., Kay, K., Oliver, M., & Gallant, J. (2009). Bayesian reconstruction of natural images from human brain activity. Neuron, 63, 902–915.

108. Lee, Y., Granger, R., & Raizada, R. (2010). How categorical are brain areas processing speech? (Under review).

109. Kriegeskorte, N. (2009). Relating population code representations between man, monkey, and computational models. Frontiers in Neuroscience, 3, 363–373.

110. Torralba, A., & Sinha, P. (2001). Statistical context priming for object detection. Proceedings of the International Conference on Computer Vision, 763–770.

111. Torralba, A. (2003). Contextual priming for object detection. International Journal of Computer Vision, 53, 169–191.

112. Carroll, S. (2005). Endless forms most beautiful. W. W. Norton.

113. Garcia-Fernandez, J. (2005). The genesis and evolution of homeobox gene clusters. Nature Reviews Genetics, 6, 881–892.

114. Zuckerman, M., Kuhlman, D., Joireman, J., Teta, P., & Kraft, M. (1993). A comparison of three structural models for personality: The big three, the big five, and the alternative five. Journal of Personality and Social Psychology, 65(4), 757–768.

115. Lynam, D., & Widiger, T. (2001). Using the five-factor model to represent the DSM-IV personality disorders: An expert consensus approach. Journal of Abnormal Psychology, 110, 401–412.

116. U.S. Food and Drug Administration. (2002). FDA public health notification: Risk of bacterial meningitis in children with cochlear implants. Retrieved from http://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/PublicHealthNotifications/ucm064526.htm

117. U.S. Food and Drug Administration. (2006). FDA public health notification: Continued risk of bacterial meningitis in children with cochlear implants with a positioner beyond twenty-four months post-implantation. Retrieved from http://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/PublicHealthNotifications/ucm062104.htm

118. Sparrow, R. (2005). Defending deaf culture: The case of cochlear implants. Journal of Political Philosophy, 13(2), 135–152.

119. Petersen, R., Smith, G., Waring, S., Ivnik, R., Tangalos, E., & Kokmen, E. (1999). Mild cognitive impairment: Clinical characterization and outcome. Annals of Neurology, 56, 303–308.

120. Rivas-Vasquez, R., Mendez, C., Rey, G., & Carrazana, E. (2004). Mild cognitive impairment: New neurophysiological and pharmacological target. Archives of Clinical Neuropsychology, 19, 11–27.

This work was supported in part by grants from the Office of Naval Research and from the Defense Advanced Research Projects Agency

List of Acronyms, Abbreviations, and Symbols

Acronym	Description
BORN	branching object relation notation
bovw	bag of visual words
EM	expected maximization
CSL	cortico-striatal loop
JLC	joint localization and clustering
Knn	k nearest neighbor
SVM	support vector machine
VTV	vision for time-varying images